Ultrasound 3D Gesture Recognition

By
Hankyu Kim

Thesis presented in partial fulfilment of the requirements
for the degree Master of Engineering
at the Stellenbosch University

Supervisor: Professor Thomas Niesler

March 2018

Declaration

By submitting this thesis electronically, I declare that the entirety of the work contained therein is my own, original work, that I am the sole author thereof (save to the extent explicitly otherwise stated), that reproduction and publication thereof by Stellenbosch University will not infringe any third party rights and that I have not previously in its entirety or in part submitted it for obtaining any qualification.

March 2018

Abstract

Interest in gesture recognition systems have grown with the recent advancements in the field of virtual and augmented reality. Gesture recognition provides a flexible interface that is not bound by the hardware as traditional mouse and keyboard combination do. This flexibility is important in virtual environments where the user has to interact with 3 dimensional objects. Having a reliable gesture recognition system will allow development of an intuitive user interface where the user can work with virtual objects as they would do in the real life.

This research presents the development of a 2D beamforming microphone array for capturing 3D images, and the processing of the captured images for gesture recognition. The developed hardware consists of an 8x8 square array of MEMS microphones that capture pulsed sinusoids emitted by an ultrasonic transducer. The data from the microphone array is demodulated, filtered, and beamformed using appropriate methods to produce 3D images of the scene.

The captured data is then processed, extracting only the relevant features, to a set of time-series vectors that represents the movement of a hand – i.e. a gesture. Using dynamic time warping (DTW) and k-nearest neighbours, the presented gestures are matched with previously captured templates, thereby recognising the type of gesture that was presented. The result showed very promising outcome with 97.5% accuracy in identifying correct gestures when the gestures are presented using a reflector, and 88.2% when the gestures are presented with a bare hand.

Contents

List of Figures

List of Tables

BYLAE 1

UNIVERSITEIT•STELLENBOSCH•UNIVERSITY
jou kennisvennoot • your knowledge partner

Plagiaatverklaring / Plagiarism *Declaration*

1 Plagiaat is die oorneem en gebruik van die idees, materiaal en ander intellektuele eiendom van ander persone asof dit jou eie werk is.
Plagiarism is the use of ideas, material and other intellectual property of another's work and to present is as my own.

2 Ek erken dat die pleeg van plagiaat 'n strafbare oortreding is aangesien dit 'n vorm van diefstal is.
I agree that plagiarism is a punishable offence because it constitutes theft.

3 Ek verstaan ook dat direkte vertalings plagiaat is.
I also understand that direct translations are plagiarism.

4 Dienooreenkomstig is alle aanhalings en bydraes vanuit enige bron (ingesluit die internet) volledig verwys (erken). Ek erken dat die woordelikse aanhaal van teks sonder aanhalingstekens (selfs al word die bron volledig erken) plagiaat is.
Accordingly all quotations and contributions from any source whatsoever (including the internet) have been cited fully. I understand that the reproduction of text without quotation marks (even when the source is cited) is plagiarism.

5 Ek verklaar dat die werk in hierdie skryfstuk vervat, behalwe waar anders aangedui, my eie oorspronklike werk is en dat ek dit nie vantevore in die geheel of gedeeltelik ingehandig het vir bepunting in hierdie module/werkstuk of 'n ander module/werkstuk nie.
I declare that the work contained in this assignment, except where otherwise stated, is my original work and that I have not previously (in its entirety or in part) submitted it for grading in this module/assignment or another module/assignment.

Studentenommer / *Student number*	**Handtekening / *Signature***
Voorletters en van / *Initials and surname*	**Datum / *Date***

Chapter 1

Introduction

Gesture recognition systems provide an immersive means of human-computer interface, allowing a more intuitive control than traditional interfaces such as keyboard and mouse can provide. The advantage of gesture recognition originates from the flexibility of the gesture interface that is not bound by the hardware. Such flexibility of gesture recognition is hugely beneficial, for example, in virtual environments where the user has to interact with 3 dimensional objects. With the recent advancements in virtual and augmented reality, interest in gesture recognition systems to provide such a flexible means of input has grown rapidly. Reliable gesture recognition systems can allow more intuitive user interfaces both in the real as well as the virtual world.

Currently there are a number of commercial products that allow gesture recognition. While optical sensor based solutions such as the Microsoft Kinect and the Leap Motion Controllers are popular at the moment, there is ongoing research to find an alternative methods with which to augment or even replace the optical systems. This research aims to investigate an alternative gesture interface system that is not based on optical sensors in an effort to expand the possibilities of gesture recognition.

This report is organized as follows. Firstly, the possibilities and limitations of various remote sensing techniques are investigated and compared. Almost any sensor that can detect an object can be used to capture a gesture to some degree. The remote sensors with examples of gesture recognition implementations include: passive infrared sensors, infrared and optical cameras, ultrasound, WiFi, and other radar based systems. However, some of these require overly complicated hardware and software, while others fail to capture sufficient information for gesture recognition. The survey on sensors concludes that an ultrasonic gesture recognition provides a balance between complexity and performance, and therefore it is a promising alternative to optical systems that is worth investigating further.

The following chapter describes how an image can be obtained using ultrasound through a beamforming array and the ways to improve the image quality. Mathematical models are derived for the beamforming techniques,

and are then used to design hardware and software for ultrasound imaging. Chapter 4 and Chapter 5 implements the beamformer designed in Chapter 3 into a system that is capable of capturing gestures. The design is tested using simulated signals to confirm the accuracy of the models and the implementations. Subsequently, the ultrasound imaging system is used to capture a series of gestures in Chapter 6.

In Chapter 7, the captured images are processed for classification of the gestures. The raw data collected from the microphone array is unnecessarily huge and extremely inefficient to work with. Feature extraction extracts only the necessary data about the gesture, hence reducing the data and simplifying the classification algorithm. The similarity between the extracted features are scored and used to classify the presented gesture based on their similarity towards predetermined templates.

The development of ultrasound based 3D imaging system for gesture recognition has been attempted previously (Przybyla *et al.*, 2014). However, the research focused only on the development of hardware and further processing for classification of gesture was not presented. The research presented here shows the methods and challenges involved in developing a gesture recognition system from a mathematical model, to an implementation in hardware, software, and application. While each of the steps in producing and processing the gestures can still be improved, this is the first attempt so far to develop such a system from end to end. At the end, the developed system will be used to capture a set of gestures presented in a 3D space and the gestures will be identified.

Chapter 2

Sensors for gesture recognition

Gesture recognition may be broken down into 5 steps: 1) the presentation of the gesture, 2) gesture capture, 3) motion capture, 4) feature extraction and 5) classification (Berman and Stern, 2012). The choice of sensors affects steps 2) and 3), which are responsible for capturing movements of a user. The type of sensor used also effects the capturing process in many ways. For example, some systems require sensors to be attached to the user to capture gestures. However, the focus of our research will be limited to systems that are free of such on-body instruments. Any sensor that can remotely detect the presence of an object and its movement may be used, but each has different advantages and limitations. In the following sections, different sensors applicable to gesture capturing are reviewed and our eventual choice is motivated.

2.1 Optical sensors

Most currently available gesture recognition systems are based on optical imagery. These systems capture a gesture using one or more cameras and process the captured images with computer vision algorithms to identify the gestures performed. Optical imagery offers high resolution that can be used to identify fine detail such as individual fingers at a distance, which is currently unmatched by other sensing mechanisms.

Examples of optical sensor based gesture recognition include the Kinect by Microsoft, the Leap Motion Controller, and many others. One drawback of optical sensors is that, when used individually, they do not provide depth information. While it is possible to identify gestures without the depth data of an image, it is undoubtedly advantageous to have information in an extra dimension. Therefore current hardware includes different ways of optically measuring distances. For example, Kinect uses time-of-flight based depth sensitive cameras, that can measure the time light takes to return from the target. Meanwhile, Leap Motion Controllers use stereoscopic imaging to create a depth sensitive image based on the difference of two images captured simultaneously

from two cameras.

While the optical sensors are used most widely, they have a few fundamental limitations. Firstly, they have a limited field of view (FOV) and require an unobstructed view of the target. Secondly, the lighting conditions of the environment can significantly effect the performance of the device. This makes it very difficult for optical techniques to be used, for example, in a moving car. Lastly, the computational requirements of analysing the high resolution images in real time are very high. Optical sensors are currently being used for gesture recognition, but the above limitations can lead the users to experience the systems as slow and unreliable. Therefore, this research aims to explore an alternative method for gesture recognition that can overcome some of the limitations of optical sensors.

2.2 Passive Infrared (PIR)

PIR sensors represent the lower end of gesture recognition systems in terms of quality and performance. However, they have the attraction of a very low cost. PIR sensors detect the intensity of infrared radiation received by the sensor. A human body radiates infrared radiation in the form of body heat, and this can be detected by a PIR sensor as a change in the intensity of the incoming infrared radiation. However, the intensity alone is not sufficient to perform gesture recognition, and hence it is most commonly used in alarm systems where the specifics of a movement are not important. However, by using multiple PIR sensors and triggering them in a predefined order, simple gestures such as a wave of a hand can be performed (Wojtczuk *et al.*, 2012). Such passive infrared sensors offer a very low cost solution in terms of both material and computational complexity, at the price of limited accuracy and functionality. Therefore PIR sensors are being used in applications where an accurate identification of the gestures is not key, such as the playback control on a Bluetooth speaker.

2.3 Radar

Radar (Radio Detection and Ranging) is a versatile sensing technique that has been used extensively for the remote detection of various objects. Radar can measure distance and velocity of an object quickly and efficiently. Hence tracking distant objects such as aircraft, sea vessels, and land vehicles has been the primary application of the radar. Gesture recognition can be considered as an application of such a tracking radar. It can however also be considered as an application of radar imaging.

A basic tracking radar sends out pulses of a known signal and analyse the returned echo's power, frequency shift, and time delay to gather information

on the target's movement. In terms of gesture recognition, a tracking radar can be used to follow the movement of hands or other body parts, and these motion tracks are subsequently used to recognize gestures. As a tracking radar does not give us the information on the orientation or the shape of the target, this method cannot be used for identifying subtle differences in gestures such as a change in shapes of the hand. However, it is still sufficient for recognising simple and large gestures. The main advantage of radar tracking based gesture recognition is its simplicity when compared to radar imaging.

Radar imaging, on the other hand, produces a 2D or 3D image of a scene using radar. The captured image can be processed in a similar way as an optical image for the purpose of gesture recognition. This will allow more intuitive and in-depth analysis of the gesture compared to a tracking radar. The obvious trade-off is an increase in complexity of both hardware and software when images are created with radar.

Generally speaking, the angular resolution θ of radar imaging is proportional to the wavelength λ and inversely proportional to its aperture length D as shown in the Equation 2.1.

$$\theta \approx \frac{\lambda}{D} \tag{2.1}$$

However, shorter wavelength in general requires more sophisticated hardware that can be vastly different from the typical radars, especially when the frequency reaches hundreds of GHz and becomes a millimetre-wave. As the design of radar depends heavily on the wavelength relative to its target, the discussion on the practicality of radar on gesture recognition is separated into two parts.

Due to differences in the properties of radar operating at different frequencies, they are divided into a number of bands as shown in the Table 2.1. Assuming the target is few centimetres in diameter, C-band and lower bands have wavelengths longer than the target while the wavelength of X-band and above would be shorter than the target.

2.3.1 C-band and lower

The C-band occupies frequencies between 4 and 8 GHz in the electromagnetic spectrum, corresponding to wavelengths of 75 mm and 37.5 mm respectively. As the average width of a human male's palm is 84 mm (TheAverageBody.com, 2014), the wavelength of the C-band radio wave is just shorter than a human hand and the wavelength of the lower frequency bands would exceed the palm's width. Gesture recognition using radar systems operating at these relatively low frequencies have gained attention in some recent studies. While the long wavelength is inadequate for detailed imaging, these bands have the advantage that they are widely used for communication and therefore hardware is readily available. For example, Wi-Fi signals operate either at 2.4 GHz (802.11 b/g/n) or 5 GHz (802.11 n/ac) depending on the standard used. Utilising these communication signals as sources for radar allows the design of a passive

Band designation	Frequency range (GHz)
HF	0.003 to 0.03
VHF	0.3 to 1
UHF	1 to 2
L	1 to 2
S	2 to 4
C	4 to 8
X	8 to 12
Ku	12 to 18
K	18 to 27
Ka	27 to 40
V	40 to 75
W	75 to 110
mm	110 to 300

Table 2.1: IEEE standard radar-frequency bands

system that does not require a dedicated transmitter (Kellogg *et al.*, 2014; Zhao *et al.*, 2014; Nandakumar *et al.*, 2014; Tang *et al.*, 2015). However, all passive detection systems based on commercial Wi-Fi hardware are based only on the signal power measurement, as the devices are not equipped to perform more complex signal processing, such as Doppler frequency measurement (Abdelnasser *et al.*, 2015).

Even when commercial hardware is not used, the Wi-Fi band may be convenient for motion tracking as it is license free. When hardware that is capable of more complex modulation is used, it was shown that the Wi-Fi band can be used to achieve more complex gesture recognition. For example, through-the-wall gesture recognition (Adib and Katabi, 2013) and gesture detection with whole-house coverage (Pu *et al.*, 2013) were demonstrated using software defined radio (SDR) operating on Wi-Fi bands. Even an attempt to create an image using a Wi-Fi band signal has shown success (Huang *et al.*, 2014). However, due to the long wavelength, the achieved image resolution was limited. Even with a 8×8 array of receivers used in Huang's experiment, the image had an error margin of 5 to 8 cm. From the above studies, it can be concluded that low frequency radar systems have potential when gestures must be recognised over a wide area, but are less suited for higher resolution applications such as hand gesture recognition.

2.3.2 X-band and above

The X-band begins after the C-band and occupies frequencies between 8 and 12 GHz. The use of higher frequency bands in radars produce superior angular and range resolution, while maintaining mobility of the system. Therefore

these frequency bands are popularly used in many radar systems. The disadvantage of using an higher frequency electromagnetic wave is that they suffer more attenuation, making the radar's effective range shorter and unsuitable for applications such as through-the-wall detection that require the signal to penetrate solid objects. However, being able to make more precise measurements and therefore higher resolution images may be key for more detailed gesture processing.

Current efforts in using high frequency radar for gesture recognition include the use of a 60 GHz mono-pulse FMCW radar for hand gestures, complemented by other sensors (Molchanov *et al.*, 2015) and Google's project Soli which is also using a band around 60 GHz frequency (Tyson, 2015). Although they are still under development, the proclaimed performance is outstanding when compared to other gesture recognition systems. With even higher frequencies in the millimetre-wave band, it is possible to achieve sub-millimetre image resolution (Cooper *et al.*, 2008). The short wavelength also allows an extremely dense array to be packed on a chip, reducing footprint of the radar imaging hardware (Arbabian *et al.*, 2013). However, working with the millimetre wave involves expensive hardware and extremely complex circuit design. While such systems are more than capable of 3D imaging for hand gesture recognition, the cost and technical complexity of the development makes it unviable for this project.

2.4 LIDAR

LIDAR is a remote sensing method that uses light emitted by a laser for target detection and ranging. LIDAR is well-suited for the precise tracking of objects in 3D, and hence it can be used for gesture recognition in high detail (Perrin *et al.*, 2004; Cassinelli *et al.*, 2005). However, LIDAR is not as efficient for imaging, as a laser is naturally extremely focused. Due to this property of a laser, an image must be produced with LIDAR through the sequential measurement of individual points. For example, a state-of-the-art LIDAR imaging device is reported to only create 15×11 pixel image at a 15 Hz frame rate. While higher resolution images are possible, increasing the resolution to 255×191 pixels reduces the frame rate to 0.06 Hz, which would be too slow to track any moving object in realistic situations (Riu and Royo, 2013). In comparison, typical optical image based gesture recognition systems such as Kinect readily offer 640×480 pixel resolution at 30 frames per second. Therefore, LIDAR imaging technology is still premature for use in gesture recognition. Rather, LIDAR is often used to complement an image based system for distance measurement.

2.5 Ultrasound

All sensors considered so far use electromagnetic (EM) signals for detection. Sound is an alternative medium that can also be used for remote sensing. Ultrasound is specifically often used as a proximity sensor for cars and alarms to detect objects, as well as for medical imaging. Because sound propagates as a wave, many of the same remote sensing methods are directly applicable. However, because sound travels in a different medium from light, it has different properties that could be advantageous in certain applications.

Sound travels much slower in air than light, which is the most notable difference between the sound wave and electromagnetic wave. Since $f = \frac{v}{\lambda}$, the slow propagation speed v of sound causes the frequency f to be lower than that of an electromagnetic wave with an equivalent wavelength λ. Lower frequency signals have the advantage, for example, of being possible to sample at the baseband without a demodulation stage, therefore simplifying the hardware. When compared to optical image based gesture recognition, ultrasound is an attractive solution which can produce a high range and angular resolution while requiring much less power and computation (Przybyla *et al.*, 2014).

Another advantage of ultrasound is that it is possible to use consumer grade speakers and microphones. For example, gesture recognition has been demonstrated using the built in speakers and microphones of a laptop without the need for any additional hardware (Gupta *et al.*, 2012). The limitation of consumer grade hardware is that it is usually designed to work only in the audible audio range and therefore are not effective when applied to ultrasound. Despite this limit, being able to use the off-the-shelf hardware is a big advantage of ultrasound.

However, ultrasound is currently not a popular choice for high resolution imaging for gesture recognition. This is partly due to the slow-propagation and high attenuation of sound in air (Moebus and Zoubir, 2007). Also, it is not possible to perform through-the-wall detection with acoustic waves (Hunt *et al.*, 2001) which is the chief advantage of long wavelength radar based gesture recognition. Also, the attenuation of sound in air increases rapidly as the frequency increases, therefore limiting the range achievable at the high frequencies needed for better image resolution (van Willigen *et al.*, 2014). Finally, the speed of sound is dependent on the temperature, humidity and pressure of the air, which could reduce the accuracy of detection (Berman and Stern, 2012).

Despite these limitations, the low-cost hardware and relative ease of development makes ultrasound an attractive method for gesture recognition. Resolution limits can be improved by using multiple sensors and better signal processing, and this in turn is possible due to the lower hardware cost and reduced computation.

2.6 Conclusion

The aim of this chapter was to identify a sensing method that could overcome some of the limits of optical image based gesture recognition systems that currently dominate the market. Initially, radar based gesture recognition appeared to be an attractive solution. Radar imaging has the potential to produce high quality images and some of them have advantage of being able to detect gestures through obstruction. However, it was discovered that, in order to develop a radar imaging system with a resolution comparable to an optical system, millimetre-wave radar had to be used. Such hardware is both expensive and difficult to develop, and the associated signal processing is computationally demanding.

Ultrasound offers many advantages over radio-waves. Most importantly, the hardware required for ultrasound imaging is much cheaper and simpler than the radar counterpart at comparative wavelengths. While it has drawbacks such as high attenuation in air and slow propagation speed, the high attenuation is less of a drawback in a short range application and the slow-propagation could be advantageous as it lowers the required sampling rate. Furthermore, recent development in hardware has allowed the low-cost development and processing of ultrasound array. Other remote sensing mechanisms are either too slow (LIDAR) or too imprecise (PIR) to be used for producing high resolution image that we aim to achieve. In conclusion, the study of sensors shows that ultrasound appears to be the most suitable sensing mechanism to be used for gesture recognition in this project.

Chapter 3

Remote sensing with ultrasound

In the previous chapter, various types of sensors that could be used for gesture recognition were compared and ultrasound was chosen as the most suitable sensor. This chapter describes how ultrasound can be used for remote sensing and gesture recognition with a focus on beamforming and three-dimensional imaging, to give a theoretical background for further development of the research.

Acoustic imaging can be implemented through the use of sonar techniques. Sonar may refer to different applications of sound such as navigation, communication, and object detection. Gesture recognition using ultrasound is an application of sonar for object detection and tracking. Sonar can be implemented as active sonar that detects the echoes of the transmitted sound, or passive sonar that relies on signals produced by external sources. This research only focuses on active sonar imaging as the body parts we are interested in detecting do not generate acoustic signals of their own. Simple ultrasound range-finder can compute the distance, approximate size and the longitudinal velocity of an object respectively from the time an echo takes to return, the echo's intensity, and the frequency shift caused by the Doppler effect. The ways in which imaging can be achieved with ultrasound, the variables effecting the performance and the techniques used to improve the systems are discussed further below.

3.1 Ultrasound imaging

The simplest type of sonar, that consists of a single pair of fixed source and receiver, is only able to detect distance of an object and its radial movement towards or away from the sonar. To create an image, the sensor must be able to identify the returned signal's direction of arrival. This is achieved by sweeping through a range of solid-angles covering the desired scene and mapping the gathered information to an image. The performance of such an ultrasound imaging system is quantified by the angular resolution of the created image.

This resolution is given by the smallest angle for which two point-like objects are distinguishable. Multiple definitions of angular resolution can be found in the literature. In this study, two objects are taken to be distinguishable when the power of reflected signal from an off-focus object is less than half of the object in focus.

It is important to first define the orientation and axis used in this work to ensure consistency. In this research, a spherical coordinate system is used with θ indicating azimuth angle measured from X axis on X-Y plane, while ϕ indicates incident angle measured from Z axis. Signal is always assumed to be originating from the positive Z axis and the sonar hardware is assumed to be on the X-Y plane. Figure 3.4 shows an example of the axes and the angles used in this research in 3D space. For 2D projection, the X-Z plane with Y axis removed will be used to maintain consistency in the direction of objects and the sonar.

Figure 3.1 shows the 2D radiation pattern on X-Z plane of a typical real-aperture transducer located at the origin and directed toward the Z axis. The radiation pattern is a frequency dependent function that describes the amplitude of a transducer's field strength, as a function of the incident angle of the signal source. The signal power of a signal from such a transducer is equal to the square of the signal's amplitude. The terms directivity pattern, antenna pattern, and beam pattern are used in the literature to refer to both the radiation pattern and the power pattern, thus it may cause confusion. The difference is trivial, especially when logarithmically scaled, as the power plot is produced by doubling the magnitude of a field plot.

Radiation patterns are reciprocal, meaning the pattern is identical when used for both transducers and receivers. The angle between two points in which the radiated power falls below 3 dB of the maxima, as shown in Figure 3.1, is referred to as the half-power beamwidth. Since the angular resolution corresponds to the angle from the maximum to the 3dB boundary, the angular resolution is simply the half of the half-power beamwidth. Consequently, a directional transducer with high gain and a small beamwidth is needed to create a high resolution image. Hence the design of an ultrasound imaging system involves the designing of a high gain transducer with the capability of manipulating the focus in order to sweep the scene.

While many variables affect the gain and beamwidth of a transducer, it depends most strongly on the size of its aperture (McCowan, 2001). Aperture refers to the physical area within the transducer that transmits or receives a propagating wave. For an acoustic transducer, aperture is the physical diaphragm that converts electric signals to and from acoustic energy. For a transducer, each point on the aperture can be viewed as a point source, if the aperture's size is comparable to the signal's wavelength. The output of the transducer is then obtained by integrating the signals emitted by the point sources. This phenomenon is known as the Huygens-Fresnel principle. For a 1D aperture in a 2D setting, the wave exhibits an interference pattern caused

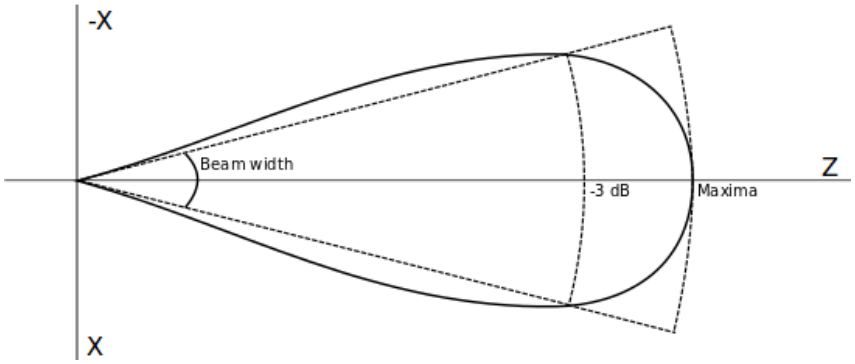


Figure 3.1: An example of the radiation pattern $R(\theta, \phi)$ of a real-aperture.

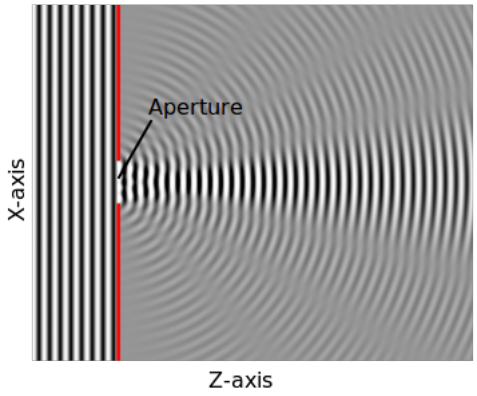


Figure 3.2: Diffraction from a single slit (Lyon, 2007).

by single slit diffraction as illustrated in Figure 3.2. For a 2D aperture, the diffraction pattern is modelled using Rayleigh-Sommerfeld diffraction formulas (Franco *et al.*, 2011). The Rayleigh-Sommerfeld model allows the accurate calculation of the directivity pattern. However, due to the complexity of solving the Rayleigh-Sommerfeld integral equations, an approximation is often used instead.

Many parameters have been developed to quantify the directivity of a transceiver without solving the Rayleigh-Sommerfeld model. A directivity index D_{ind} is the ratio of the maximum power to the average power of a radiation pattern, and it is defined as:

$$D_{ind} = \frac{1}{\frac{1}{4\pi}\int_0^{2\pi}\int_0^{\pi} |R(\theta,\phi)|^2 \sin(\phi) d\phi d\theta} \tag{3.1}$$

Where $R(\theta, \phi)$ is the normalised radiation pattern. The smallest possible value of D_{ind} is 1, which means the transducer is isotropic and radiates evenly in all directions. If the directivity index is bigger than one, the power of the

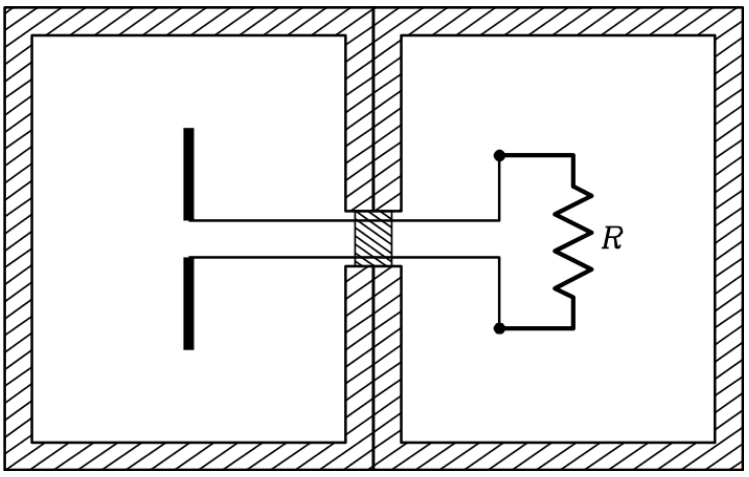


Figure 3.3: Two cavities in a thermal equilibrium (Condon and Ransom, 2016).

signal from a particular direction will be D_{ind} times more than the same signal received by an isotropic transceiver.

Directivity is not the only factor of an aperture that effects the signal power. Any aperture intrinsically loses some of the power as it transforms the energy. The ratio of the total input power to the total output power is referred to as the radiation efficiency ξ_{rad}. Gain is used to describe the ratio of input power to the output power, by combining the directivity index and the radiation efficiency.

$$G = \xi_{rad} D_{ind} \tag{3.2}$$

Similarly to the directivity index, $G = 1$ indicates an isotropic radiator with perfect radiation efficiency (no loss).

Finally, the gain can be related to the effective aperture. Effective aperture refers to the area that is needed to capture a signal to a desired strength, given a field with flux density S measured in W/m^2. The received power $P_r(f)$ is therefore:

$$P_r(f) = A_e S \tag{3.3}$$

where A_e is the effective aperture in m^2. The effective aperture A_e can be calculated from the gain by using the Equation 3.4 (Stutzman, 1998).

$$A_e = \frac{\lambda^2}{4\pi} G \tag{3.4}$$

To understand this equation, one can think of two cavities next to each other, in a thermal equilibrium as shown in Figure 3.3. One cavity contains a resistor and the other contains a receiver. If the resistor and the receiver are connected, the received power by the two components should be equal for the system to not violate the second-law of thermodynamics. The only source of power for the resistor is the thermal noise, which is equal to $k_B T B$ where k_B is the Boltzmann's constant, T is the temperature in kelvin and B is the

frequency bandwidth of the signal. On the receiver's side, the captured power originates from black-body radiation from its environment, and the flux of this radiation can be modelled using Rayleigh-Jeans approximation. Therefore, the power received over a narrow bandwidth B can be approximated by Equation 3.5. Note the flux is halved to account for only the signal with a matched polarisation.

$$\begin{aligned} P_r &= A_e S B \\ &= \frac{B}{2} \int_0^{2\pi} \int_0^{\pi} A_e \frac{2k_B T}{\lambda^2} \sin(\phi) d\phi d\theta \end{aligned}$$

$$P_r = \frac{k_B T B}{\lambda^2} \int_0^{2\pi} \int_0^{\pi} A_e \sin(\phi) d\phi d\theta \tag{3.5}$$

$$P_r = \frac{4\pi k_B T B}{\lambda^2} A_e \tag{3.6}$$

Assuming the receiver is isotropic, and therefore has constant effective aperture, the Equation 3.5 can be simplified to Equation 3.6. Equating the received power and the thermal noise gives us the effective aperture for an isotropic receiver (Condon and Ransom, 2016).

$$\begin{aligned} k_B T B &= \frac{4\pi k_B T B}{\lambda^2} A_e \\ A_e &= \frac{\lambda^2}{4\pi} \end{aligned} \tag{3.7}$$

As the receivers in general are not always isotropic, the effective aperture has to be scaled by the gain, leading us back to the Equation 3.4. For an electrically large antenna, the effective aperture is directly proportional to the physical aperture area and thus $A_e = \xi_{ap} A_p$ where ξ_{ap} is the aperture efficiency (Chryssomallis *et al.*, 1999). Therefore, Equation 3.4 shows that the larger aperture produces a higher gain and therefore a narrower beamwidth.

Equation 3.8 provides a simple estimate of beamwidth from directivity:

$$D \approx \frac{DB}{\theta_{3dB} \cdot \phi_{3dB}} \tag{3.8}$$

where θ_{3dB} and ϕ_{3dB} refer respectively to the half-power beamwidth in terms of azimuth and incident angle and DB refers to the directivity-beamwidth product as shown in Table 3.1 (Stutzman, 1998). The directivity-beamwidth product combines all of the factors that effect the directivity of an aperture, allowing a quick estimate of the beamwidth from the directivity or the gain. While the estimate is only accurate to within 25 percent, it confirms the original assumption that a higher gain equates to a smaller beamwidth (Johnson and Jasik, 1993).

Aperture type	Directivity-Beamwidth Product
Uniform rectangular aperture	32383
Uniform circular aperture	35230
General use for practical antennas	26000

Table 3.1: Directivity-beamwidth product for simple geometric apertures.

From the above relationships, it can be concluded that in order to create a better image, one can either increase the size of the aperture or increase the frequency. Operating frequency is often predetermined and fixed by the hardware used and hence the size of the aperture is the only parameter that can be changed to improve the image resolution within the design. However, constructing a physically large aperture to achieve a more focused beam is expensive. A mechanical system can be used to move or rotate the transducer in order to sweep the beam. However, mechanical controllers are bulky, expensive, slow, and difficult to control accurately. To avoid these drawbacks, alternative ways of synthesising a large aperture have been developed.

3.2 Sound propagation model for an array aperture

In order to increase the effective aperture, an array of elements can be used together as a single transceiver. This process of combining the signals from an array to achieve better focusing is known as beamforming. An array of microphones allows the design of a highly directional microphone without the need to change the frequency or construct a physically large aperture. Furthermore, the radiation pattern of the array can be manipulated by controlling the gains of individual microphones before the signals are combined. In this way physical movement of the receivers becomes unnecessary. This gives a beamforming array an advantage over a physical aperture in design and operational flexibility.

In order to design a beamforming array, we first need to understand how the sound is propagated to individual microphones. Figure 3.4 shows the 3 dimensional space that will be considered, with the orientation of the microphone array marked by grey rectangles. The microphones are laid out in the X-Y plane and the signal they are receiving is assumed to be coming from the positive Z direction.

The reciprocity of a radiation pattern allows us to treat transmitter and receiver beamforming in the same way. Indeed, often beamforming arrays are used for both transmission and reception in order to maximise the focus of an ultrasound imager. Using arrays at both ends can also be used to cancel the effects of side-lobes, by aligning the side-lobes of the one with the nulls of the other. Side-lobe cancelling significantly boosts the gain and hence

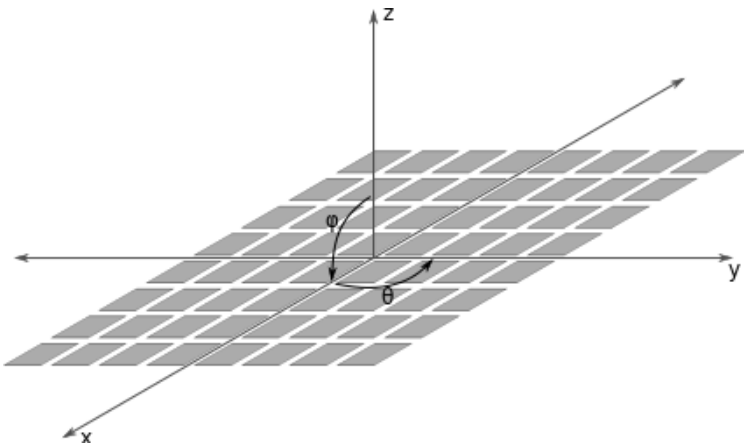


Figure 3.4: Orientation of the 2D microphone array in a 3D space.

the image resolution of the system (Harput and Bozkurt, 2008). However, because transmission to different focal points cannot be performed simultaneously, each three dimensional pixel must be probed individually. Combined with the slow propagation speed of sound in air, it leads to a slow refresh rate of the ultrasound imaging device. The delay can be reduced by transmitting from one transmitter at a time, and synthesising the transmitter beamforming in software (Dokmanic and Tashev, 2014). This makes the number of transmissions required per image to be equal to the number of transmitters, not the number of pixels to be imaged. However, the solution still requires multiple transmissions per image and significantly more storage and computation for the beamforming synthesis. Moreover, when compared to using an array of microphones, the transmitters need a larger footprint and typically require more complex hardware. Hence the advantage in gain by beamforming in both ends is diminished. Therefore, only the receiver beamforming with a single transmitter will be used for this study.

The signal received at an aperture needs to be modelled differently in the near-field (known as the Fresnel zone) and in the far-field (known as the Fraunhofer zone). A near-field signal experiences complex diffraction and interference that is difficult to model. The distance at which the far-field begins is not precisely defined and varies in the literature. It depends on the definition of far-field and the requirements for each application (Capps, 2001). However, generally speaking, and for our application, it is sufficient to note that the region becomes a far-field if Equation 3.9 is satisfied for the distance from the aperture d, where D is the aperture diameter and λ is the wavelength of the signal(Kino, 1987; Ng and Swanevelder, 2011).

$$d \gg \frac{D^2}{\lambda} \tag{3.9}$$

In our case the ratio of the diameter of each microphones' aperture to the wavelength is sufficiently small for us to assume that the far-field assumption

for each microphone is always valid (Shung and Zippuro, 1996).

A far-field source can be viewed as a point source $s(t)$, radiating spherically. Such radiation from a position $\boldsymbol{p_s}$, received by the microphone m, can be modelled as follows:

$$\begin{aligned} x_m(t) &= d_m(t, \boldsymbol{p_s}) * s(t) + n(t) \\ &\mathcal{F} \downarrow \\ X_m(f) &= D_m(f, \boldsymbol{p_s}) S(f) + N(f) \end{aligned} \tag{3.10}$$

In these equations D_m represents the distortion on the signal due to the propagation. Assuming the microphone itself is omnidirectional and there is no preamplifier distortion, the received signal experiences only a delay and attenuation as shown in Equation 3.11, where $\boldsymbol{p_m}$ is the position vector of the microphone.

$$D_m(f, \boldsymbol{p_s}) = \frac{1}{|\boldsymbol{p_s} - \boldsymbol{p_m}|} e^{-j\frac{2\pi f}{v}|\boldsymbol{p_s} - \boldsymbol{p_m}|} \tag{3.11}$$

The goal of the array aperture is to recover the original signal $s(t)$ from Equation 3.10 by removing the noise $n(t)$ and reversing the effect of the distortion.

Like individual microphones, the array as whole also has separable near-field and far-field regions. The regions effect how the received signals vary among the microphones in the array. Equation 3.11 is modelled for the microphones' far-field, and can be used regardless of which region is relevant regarding the array. However, it is possible to simplify the model further for a signal originating from the array's far-field region. A spherical wave has a diminishing curvature as the radius increases. Hence, as the radius increases, it begins to approximate a planar wave. If all sources are assumed to originate from sufficiently far away, we can ignore differences in attenuation using far-field approximation. Phase differences are however introduced when the incident angle ϕ is non-zero, as shown in Figure 3.5 as an example. Note the X axis in Figure 3.5 is aligned with the received signal's azimuth angle θ. Clearly, the phase difference $\Delta\Phi$ is related to the microphone's distance away from the origin along the azimuth angle and the incident angle. Using simple trigonometry, we can compute the phase difference as:

$$\Delta\Phi = \frac{2\pi f}{v}(\boldsymbol{p_m} \cdot 1\angle\theta)\sin(\phi) \tag{3.12}$$

This allows Equation 3.11 to be simplified to Equation 3.13. This equation, which computes the delay experienced by each microphone, models sound propagated from a far-field source and forms the basis of the phased array design.

$$D_m(f, \theta, \phi) = e^{-j\frac{2\pi f}{v}(\boldsymbol{p_m} \cdot 1\angle\theta) sin(\phi)} \tag{3.13}$$

3.3 Array pattern

Based on the propagation model to each microphone elements, an array pattern is derived to describe the effect of a beamforming array on the gain pattern. In order to derive the effect of an array receiver, we begin by analysing a beamforming array with two omnidirectional microphones placed next to each other. We can assume first microphone to be placed in the origin and calculate the effect of a second microphone additionally placed next to it as illustrated in Figure 3.5. In this example, the azimuth angle of the received signal is assumed to be zero, hence limiting the analysis to 2D space in X-Z plane. As illustrated in Figure 3.5 the incident wave travels different distances to reach each of the microphones. Assuming the incoming signal to be a planar wave, the phase difference between the two microphones $\Delta\Phi$ can be calculated as shown in Equation 3.14 for an incident angle ϕ. This conforms to Equation 3.12 with azimuth angle θ set to be zero.

$$\Delta\Phi = \frac{2\pi f}{v} d \sin(\phi) \tag{3.14}$$

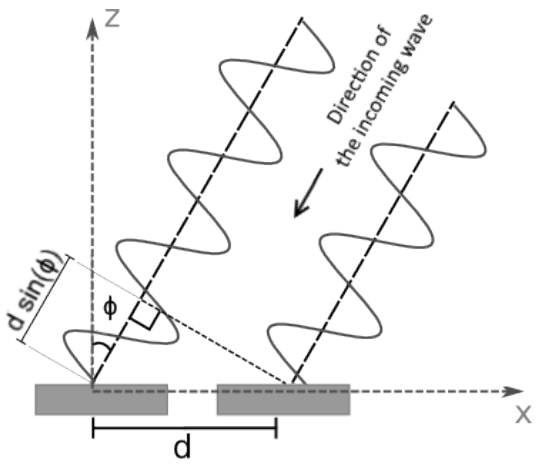


Figure 3.5: A two element microphone array, indicating the relative delay between the measured signals for an incident planar wave.

The question of an appropriate choice of the distance d between the microphone elements now arises. On the one hand we wish to maximise d since this maximises the aperture. On the other hand, d should be small enough so that the phase difference at the two microphone can be unambiguously determined. Assuming that the incident angle ϕ ranges between $-\frac{\pi}{2}$ and $+\frac{\pi}{2}$ radians, this is accomplished by ensuring that the phase difference $\Delta\Phi$ varies between $-\pi$ and $+\pi$ over this range. By making the distance between the microphone elements $d = \frac{\lambda}{2}$, the phase difference will be $\pm\pi$ when the incident angle is $\pm\frac{\pi}{2}$, thereby

exactly satisfying the criteria. Therefore, $d = \frac{\lambda}{2}$ is the best choice of separating distance between two microphone elements (Monzingo *et al.*, 2011). The resulting gain pattern of choosing $d = \frac{\lambda}{2}$ is shown below in Figure 3.6a. The figure shows a single maxima when the incident angle is zero and nulls at the incident angles $\pm\frac{\pi}{2}$ radians, showing a well matched beamforming response.

Increasing number of elements in an array improves the beamforming effect of the array receiver. Based on the far-field model, the signals received by each element of a linear array with equidistant transducers will have linearly varying phase. The received signals at the elements with linearly increasing phase can be represented as multiples of a vector $1\angle\Delta\Phi$. Using the formula to calculate the sum of a geometric sequence, the array pattern AP for a N-element array can be calculated as shown in the Equation 3.15.

$$\begin{aligned}
AP_2(\Delta\Phi) &= \frac{1}{2}|1\angle 0 + 1\angle\Delta\Phi| \\
AP_3(\Delta\Phi) &= \frac{1}{3}|1\angle 0 + 1\angle\Delta\Phi + 1\angle 2\Delta\Phi| \\
&\vdots \\
AP_N(\Delta\Phi) &= \frac{1}{N}\left|\sum_{n=0}^{N} 1\angle n\Delta\Phi\right| \\
AP_N(\Delta\Phi) &= \frac{1}{N}\left|\frac{1-(1\angle\Delta\Phi)^N}{1-1\angle\Delta\Phi}\right| \\
AP_N(\Delta\Phi) &= \frac{1}{N}\left|\frac{1-(1\angle N\Delta\Phi)}{1-1\angle\Delta\Phi}\right|
\end{aligned} \tag{3.15}$$

Equation 3.15 can be further simplified by representing the vectors using complex exponentials. This allows us to factor out the phase part of the equation which we can ignore, since we are only interested in the magnitude of the array pattern. The resulting formula to calculate array pattern of a N element linear array is shown in Equation 3.16.

$$\begin{aligned}
AP_N(\Delta\Phi) &= \frac{1}{N}\left|\frac{1-e^{jN\Delta\Phi}}{1-e^{j\Delta\Phi}}\right| \\
&= \frac{1}{N}\left|\frac{e^{j\frac{N\Delta\Phi}{2}}(e^{-jN\Delta\Phi/2}-e^{jN\Delta\Phi/2})}{e^{j\frac{\Delta\Phi}{2}}(e^{-j\Delta\Phi/2}-e^{j\Delta\Phi/2})}\right| \\
&= \frac{1}{N}\left|\frac{\sin(N\frac{\Delta\Phi}{2})}{\sin(\frac{\Delta\Phi}{2})}e^{j\frac{(N-1)\Delta\Phi}{2}}\right| \\
&= \frac{1}{N}\frac{\sin(N\frac{\Delta\Phi}{2})}{\sin(\frac{\Delta\Phi}{2})}
\end{aligned} \tag{3.16}$$

Equation 3.16 is the array pattern of an N-element linear phased array. Equation 3.14 and Equation 3.16 allow us to compute the far-field gain of an

N-element linear array at a given incident angle. According to the equation, the denominator is maximised and hence the gain is most suppressed when the phase difference $\Delta\Phi$ is equal to 180°. As for the two element array, it can be deduced that the ideal distance between the elements in a linear array, regardless of the number of elements involved, is equal to $\frac{\lambda}{2}$.

Figure 3.6 shows the array pattern of linear microphone arrays, with varying number of microphone elements while the distance is set to be $\frac{\lambda}{2}$. The figures show the progressive improvement in beamforming effect, which can be observed by narrower beamwidth and smaller side-lobes, with increasing number of elements used.

From the array patterns computed above, we can measure their beamwidth by locating the incident angle where the main-lobe's gain squared falls below 0.5. Typically, half-power beamwidth of a linear array is estimated using Equation 3.17 (Brown, 2012).

$$\theta_{3dB} = \frac{0.886\lambda}{L} \tag{3.17}$$

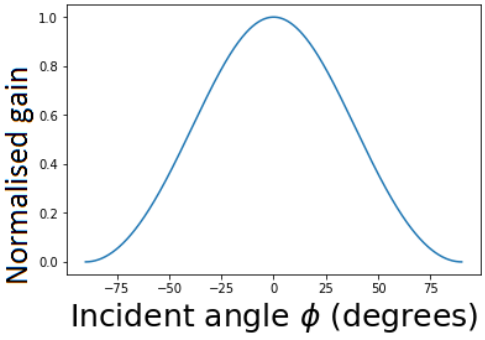
Where L and λ are the length of the array and the wavelength of the incident signal, in metres, respectively. We can assume the length of the array to be $N \times \frac{\lambda}{2}$, which leads to Equation 3.18 for the angular resolution ϕ_r.

$$sin(\phi_r) = \frac{0.886}{N} \tag{3.18}$$

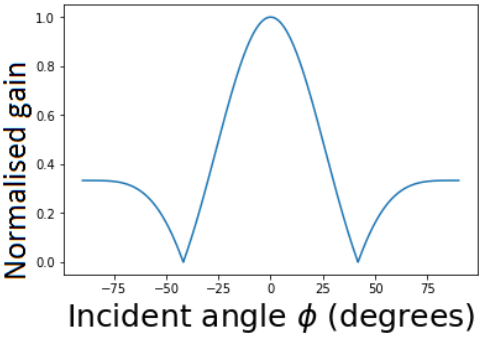
The estimated angular resolution using Equation 3.18 is compared against the angular resolution measured from the beam pattern calculated using Equation 3.16. The result of the two computations is shown in Figure 3.7. The solid blue line in the graph shows the angular resolution calculated using Equation 3.18 and the dashed orange line shows the angular resolution measured from the plot of array patterns created using Equation 3.16. The two methods give us a consistent and converging prediction of the beam resolution, thereby verifying the accuracy of the model.

While Equation 3.18 shows that a larger number of elements in an array results in a better resolution, it is important to note that the far-field boundary $\frac{D^2}{\lambda}$ increases quadratically with an increase in the number of microphones in the array. This can be shown by assuming the diameter of an aperture is equal to $N \times \frac{\lambda}{2}$ as done in Equation 3.19.

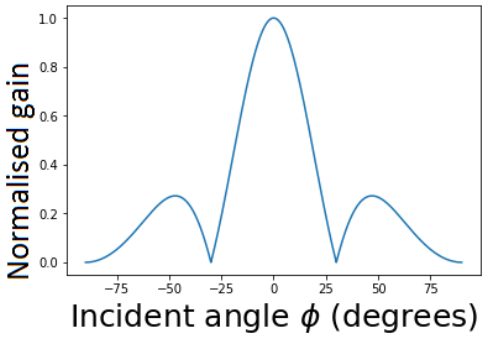
$$\begin{aligned} d &\gg \frac{D^2}{\lambda} \\ &= \frac{(N \times \frac{\lambda}{2})^2}{\lambda} \\ &= \frac{N^2\lambda}{4} \end{aligned} \tag{3.19}$$



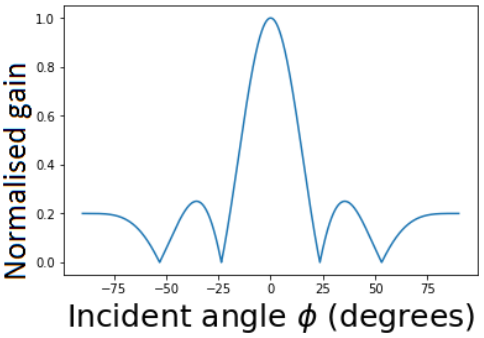
(a) 2 element array.



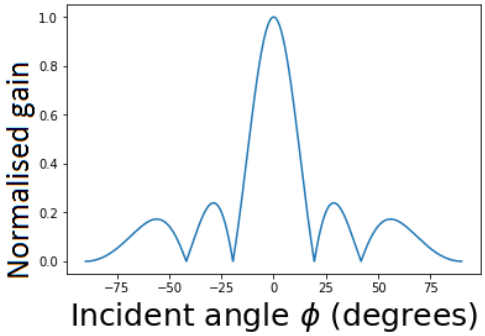
(b) 3 element array.



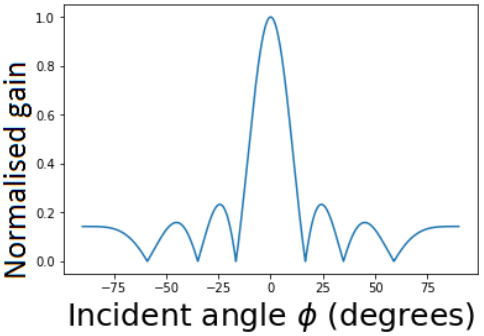
(c) 4 element array.



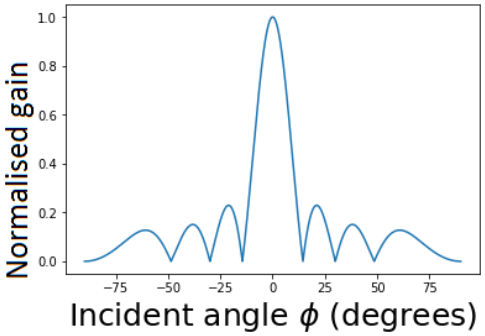
(d) 5 element array.



(e) 6 element array.



(f) 7 element array.



(g) 8 element array.

Figure 3.6: Far-field normalised gain patterns of linear microphone arrays with $d = \frac{\lambda}{2}$.

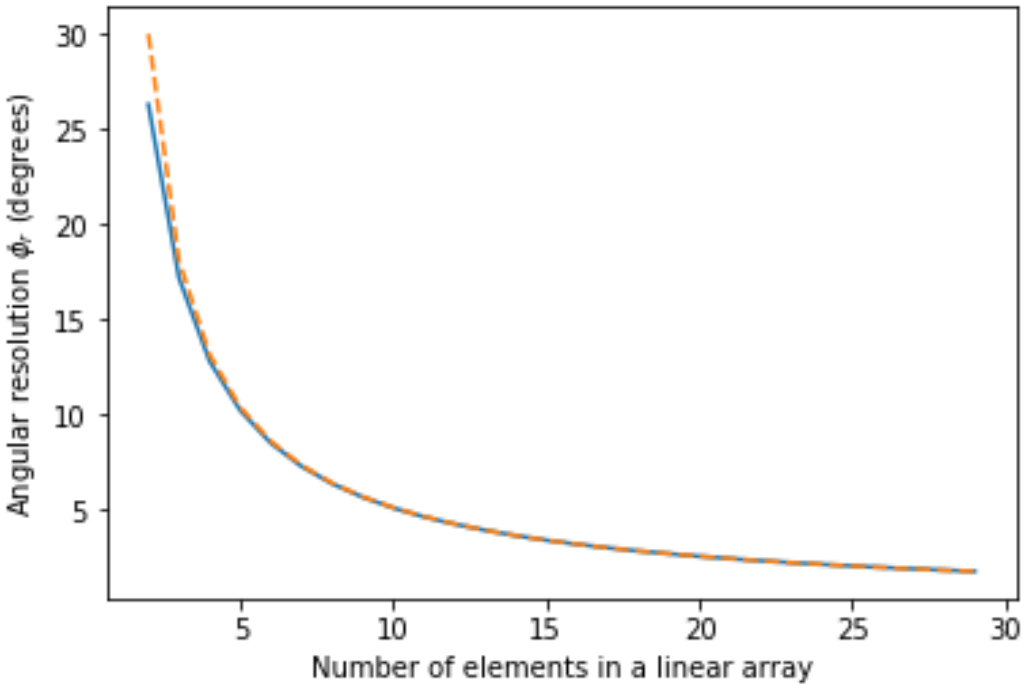


Figure 3.7: Number of elements N vs angular resolution ϕ_r for a linear phased array

Therefore, the number of elements in an array should be as many as the hardware allows, while not violating the far-field assumption. If the far-field assumption is violated, the above model for a linear array pattern becomes inaccurate and the vector based model shown in Equation 3.11 should be used instead.

3.4 Phased array beamforming

Equation 3.16 gives the array pattern of a N-element linear array, when the received signals are summed together with equal weighting. In this configuration, the beam is orientated directly ahead (orthogonal to the plane of the array). In order to steer the beam, the received signals must be weighted differently. This effectively applies a filter to the array response, and for this reason is called a filter-and-sum beamformer.

Consider a set of signals $x_n(t)$ captured by N microphones as described in Equation 3.10, and a set of N corresponding filters $w_n^*(t)$. While it is not necessary to describe the filter in a complex conjugate form now, it becomes useful later when designing adaptive filters. Therefore we define the filter as $w_n^*(t)$ in order to maintain consistency. The output of filter-and-sum beamformer for

such an array is given by Equation 3.20.

$$\begin{aligned} y(t) &= \frac{1}{N}\sum_{n=1}^{N} w_n^*(t) * x_n(t) \\ &\mathcal{F}\downarrow \\ Y(f) &= \frac{1}{N}\sum_{n=1}^{N} w_n^*(f) \cdot X_n(f) \end{aligned} \tag{3.20}$$

We can simplify the equation using matrix notation by defining:

$$\mathbf{X}(f) = \begin{bmatrix} X_1(f) \\ X_2(f) \\ \vdots \\ X_n(f) \end{bmatrix} = \begin{bmatrix} D_1(f)S(f) + N_1(f) \\ D_2(f)S(f) + N_2(f) \\ \vdots \\ D_n(f)S(f) + N_n(f) \end{bmatrix}$$

$$\mathbf{w}(f) = \begin{bmatrix} w_1(f) \\ w_2(f) \\ \vdots \\ w_n(f) \end{bmatrix}$$

Then Equation 3.20 can be rewritten as:

$$Y(f) = \frac{1}{N}\mathbf{w}^H(f) \cdot \mathbf{X}(f) \tag{3.21}$$

In Equation 3.21 $\mathbf{w}^H(f)$ indicates the Hermitian transpose or conjugate-transpose of $\mathbf{w}(f)$, thereby assuring the application of complex conjugates on the filters. Although the filters are frequency dependent, we can assume it is a constant complex coefficient $\mathbf{w}$ for a narrow band signal and refer to it as a weight vector. Using Equation 3.10 to substitute for $\mathbf{X}(f)$, we obtain:

$$\begin{aligned} Y(f) &= \frac{1}{N}\mathbf{w}^H(\mathbf{D}(f)S(f) + \mathbf{N}(f)) \\ &= \frac{1}{N}\mathbf{w}^H\mathbf{D}(f)S(f) + \frac{1}{N}\mathbf{w}^H\mathbf{N}(f) \end{aligned} \tag{3.22}$$

where $s(f)$ is the original signal we are trying to recover, $\mathbf{D}(f)$ is the propagation distortion on the signal, and $\mathbf{N}(f)$ is the random environmental noise. The weight vector should be selected such that it maximises the contribution of the array response $\mathbf{w}^H\mathbf{D}(f)$ in the first term, while minimising the contribution of the noise in the second term of Equation 3.22.

To find a suitable weight vector, we start by considering the delay-and-sum (DAS) beamformer output. DAS beamformer is formed by applying a time shift to each microphone signal to counter the delay. From the phase difference in Equation 3.14, we see that this delay must be $\frac{dsin(\phi)}{v}$. Therefore, DAS

beamforming shifts input signal $x_n(t)$ forward by the multiples of $\frac{dsin(\phi)}{v}$ and sum to get Equation 3.23 as the output. Note that in real implementations, as we cannot shift a signal forward without a priori knowledge about the signal, all received signals are delayed together as necessary to allow required advancements.

$$\begin{aligned} y(t) &= \frac{1}{N}\sum_{n=1}^{N} x_n\left(t + (n-1)\frac{dsin(\phi)}{v}\right) \\ &\mathcal{F}\downarrow \\ Y(f) &= \frac{1}{N}\sum_{n=1}^{N} X_n(f)e^{j(n-1)\Delta\Phi} \end{aligned} \tag{3.23}$$

We already know the delay experienced by the signal at each microphone from Equation 3.13. This can be represented in vector form as:

$$\mathbf{D}(f) = \begin{bmatrix} 1 \\ e^{-j\Delta\Phi} \\ e^{-j2\Delta\Phi} \\ \vdots \\ e^{-j(n-1)\Delta\Phi} \end{bmatrix} \tag{3.24}$$

The term $D(f)$ is called the steering vector or array manifold, and it describes the direction from which the signal of interest is propagating. It is obvious from Equation 3.23 that the weight vector for a delay and sum beamformer is equal to the steering vector.

$$\mathbf{w}(f) = \begin{bmatrix} 1 \\ e^{-j\Delta\Phi} \\ \vdots \\ e^{-j(n-1)\Delta\Phi} \end{bmatrix} \tag{3.25}$$

Since we choose the weight vector to counter the delay, the vectors $\mathbf{w}$ and $\mathbf{D}(f)$ consist of phasors that cancel out when the inner product is calculated. Therefore the array response for DAS beamformer is $\mathbf{w}^H\mathbf{D}(f) = N$, which from Equation 3.22 leads to the output:

$$Y(f) = S(f) + \frac{\mathbf{w}^H\mathbf{N}(f)}{N} \tag{3.26}$$

Assuming that the noise $\mathbf{N}(t)$ is uncorrelated between the microphones; the weight vector has no effect on the noise power as the multiplication is a linear operation. However, the summing and normalising by N leads to an averaging effect and reduces the noise power by the factor of N (Tashev, 2009). Hence the delay and sum beamformer theoretically has a SNR that is N times better than a single microphone.

Note, that by substituting the steering vector in Equation 3.24 into Equation 3.22 and ignoring the noise term, we find:

$$Y(\Delta\Phi) = \frac{1}{N}\sum_{n=0}^{N-1} w_n^* S_n(f) e^{-jn\Delta\Phi} \tag{3.27}$$

This has the form of a discrete Fourier transform (DFT), where $\Delta\Phi = \frac{2\pi k}{N}$. Therefore, beamforming is often viewed as a spatial Fourier transform and spatial filtering problem. The optimum distance between the elements $\frac{\lambda}{2}$ is analogous to the Nyquist sampling theorem in the time domain (Benesty *et al.*, 2008). This duality allows us to apply digital filter design principles to choose the weight vector. For example, the Dolph-Chebyshev filter can be implemented in beamforming to reduce sidelobes (Van Veen and Buckley, 1997).

3.5 Digital beamforming

Phased array beamforming can be implemented for both continuous-time and discrete-time data. Continuous-time beamforming performs the signal weighting before the signal is sampled, using analogue components. Discrete-time beamforming implements the weights after sampling. Initially, beamforming arrays were typically implemented using analogue gain and delay lines, because the high computational load required to perform beamforming digitally was impractical. However, analogue gain and delay lines are susceptible to noise and manufacturing error while also being expensive to implement. Digital computation of gain and delay is free of such errors, giving it a great advantage over an analogue implementation.

The theory of discrete-time beamforming is consistent with that of analogue beamforming. However, certain limitations exist when working with a sampled signal. A discrete-time delay line can only delay by an integer number of samples, limiting the delay resolution. The delay resolution is directly related to the steering resolution and has to be resolved. A solution involves oversampling or interpolation of the signal to a sampling rate much greater than the Nyquist rate. This however significantly increases the computational cost of the system (Mucci, 1984). To avoid the interpolation, the delay can be applied in the frequency domain. This utilises the property of Fourier transforms, which states a delay in time domain is identical to a multiplication in the frequency domain with a linearly changing phase. By applying the phase shift in the frequency domain and applying an inverse FFT, an arbitrary delay can be achieved in the time domain. The signal, however, will become complex; similar to how an quadrature modulated signal represents the phase of a signal. The two methods of applying a time domain delay are mathematically identical and therefore produce the same image quality.

Instead of time domain delay methods, frequency domain beamforming operates in the frequency domain using Fourier transformed data. As shown

in Equation 3.27, the operations of the beamforming algorithm are very similar to those of a DFT. Therefore, the FFT can be used to efficiently calculate an image from frequency domain data. Each time domain delay forms a single beam which must be scanned to produce an image. Frequency domain beamforming computes the output of the whole scene at once, greatly increasing speed. However, the frequency domain beamforming is extremely rigid, as it requires a fixed and known phase shift between the elements. This limits the application of frequency domain beamforming to complex weighted beamformers.

When it can be assumed that the received signal is narrow band, the weight vector becomes a constant. Then Equation 3.22 becomes:

$$\begin{aligned} Y(f) &= \frac{1}{N}\mathbf{w}^H\mathbf{X}(f) \\ &\qquad \mathcal{F}^{-1} \downarrow \\ Y(t) &= \frac{1}{N}\mathbf{w}^H\mathbf{x}(t) \end{aligned} \tag{3.28}$$

This means that the complex weights can be applied in the time domain, and the array output is simply a complex weighted sum of the input signals. As this is a linear arithmetic problem, the model allows us to efficiently calculate the array output. Also, adaptive algorithms that optimise the weight vector have been developed based on this model.

3.6 MVDR beamforming

Adaptive beamforming is a special kind of beamformer design that can adapt its weights to be the most suitable beamformer for a given signal. We have established in the previous section that a beamformer is an implementation of a spatial finite impulse response (FIR) filter, and therefore any adaptive FIR filter design can be applied with minimal adjustments. Wiener filtering was originally applied to FIR filtering, to minimises the mean squared value of the error signal. Hence this technique is also known as the minimum mean square error (MMSE) filter and it is known as the optimum linear discrete filter. However, the Wiener filter requires a priori knowledge of the signal's statistical properties, and it is only applicable when the input is stationary. An adaptive filter recursively corrects the filter weights, leading it to become an optimum filter for the given signal. Therefore, if the signal is stationary, the adaptive filter will converge to a Wiener filter (Haykin, 1991).

Minimum variance distortionless response (MVDR) is an adaptive beamforming algorithm developed by Capon (Capon *et al.*, 1967). It is a form of adaptive beamforming algorithm collectively called linearly constrained minimum variance (LCMV) beamformers. LCMV beamformers choose the weights such that the signals of interest are passed with a specific gain and phase -

hence the output is linearly constrained - while minimising the power of the output. Since the gain on the desired signal is already fixed, it selectively reduces the power of the noise signal. Mathematically, the LCMV algorithm is described by Equation 3.29, where g is the complex constant value specified for the array response.

$$\min_{\mathbf{w}} E(|Y(f)|^2) \qquad \text{while} \quad \mathbf{w}^H\mathbf{D} = g \tag{3.29}$$

MVDR removes distortion on the output signal by choosing the signal response $\mathbf{w}^H\mathbf{D}$ to be 1. Therefore the signal experiences no gain or phase shift by the beamformer.

$$\min_{\mathbf{w}} \quad E(|Y(f)|^2) \qquad \text{while} \quad \mathbf{w}^H\mathbf{D} = 1 \tag{3.30}$$

Notice the gain of array response is 1, meaning that the factor $\frac{1}{N}$ in Equation 3.10 has been absorbed into the weight vector. The power of the output signal can be calculated from the input signal and the array response, as follows:

$$\begin{aligned} &\min_{\mathbf{w}} \quad E(|\mathbf{w}^H\mathbf{X}|^2) \\ =&\min_{\mathbf{w}} \quad \mathbf{w}^H\mathbf{R}_x\mathbf{w} \qquad \text{while} \quad \mathbf{w}^H\mathbf{D} = 1 \end{aligned} \tag{3.31}$$

Equation 3.31 can be reformulated as a single minimisation equation using the method of Lagrange multipliers:

$$\min_{\mathbf{w}} h(\mathbf{w}) = \mathbf{w}^H\mathbf{R}_x\mathbf{w} + Re(\lambda^*(\mathbf{w}^H\mathbf{D} - 1) \tag{3.32}$$

Equation 3.32 can be solved by locating the null of its conjugate derivative:

$$\frac{\partial}{\partial \mathbf{w}} h(\mathbf{w}) = \mathbf{R}_x\mathbf{w} + \lambda^*\mathbf{D} = 0 \tag{3.33}$$

$$\text{where} \quad \mathbf{w}^H\mathbf{D} = 1 \tag{3.34}$$

This effectively turns Equation 3.31 into a set of simultaneous matrix equations, whose solution given by Equation 3.35 (Haykin, 1991).

$$\mathbf{w} = \frac{\mathbf{R}_x^{-1}\mathbf{D}}{\mathbf{D}^H\mathbf{R}_x^{-1}\mathbf{D}} \tag{3.35}$$

Equation 3.35 allows us to calculate the weights, given the steering angle of interest and the received signal. It is important to note that in order to ensure invertibility, R_X must have full rank. This requires that the input signal $\mathbf{X}$ must be independent of each other. The independence is normally assured by additive random noise in the signal. Hence the algorithm only works in the presence of random and uncorrelated noise. Since the adaptive algorithm is designed to remove uncorrelated noise, the above observation makes intuitive sense.

3.7 Expansion to a planar array

The development so far only considered a linear array working in 2D space. In order to create a 3D image, the beamforming array has to be expanded to create a planar array. This requires expanding the above calculations and computing appropriate weight vectors for the microphones in a 2D plane. We have shown that the DAS beamforming is a filter-and-sum beamforming with weight vectors chosen to be equal to the steering vector $\mathbf{D}$ in Equation 3.25. MVDR beamformer weights are also dependant only on the steering vector for a given set of signals. Therefore, in order to expand an array, we have to only compute the necessary steering vector that describes the signals received by the microphone elements in a planar array.

The propagation model for a planar array has already been developed as shown in Equation 3.13. A difference between the models for linear array and planar array is that planar array additionally incorporates position vectors of microphones instead of the scalar distance from the origin, and also the received signal's azimuth angle. So far, we have ignored the azimuth angle θ by assuming that the microphone array is aligned with the target's azimuth angle. We can begin incorporating the azimuth angle in our computation to allow phase shift beamforming in 3D space. The resulting steering vector is identical to that of the linear array as shown in Equation 3.24, except the phase difference $\Delta\Phi$ is calculated using Equation 3.12 instead of Equation 3.14 that was simplified for linear array. From the steering vector calculated with desired azimuth and incident angles, we can design DAS or MVDR beamformer that steers its focus to a direction of choice in 3D.

The resulting gain pattern of a planar array can be predicted by considering a row of the array as a single component, then a planar array is a linear array with those components placed in a column. The effect of row-wise beamforming is combined with column-wise beamforming to produce a focused beam in both axes. Because filter-and-sum beamforming with weights vector is a linear computation, the two orthogonally operating linear beamformers do not effect each other in their respective orientations. Therefore the planar array will have an identical angular resolution to a linear array with same number of elements along X and Y axis.

A signal arriving at an angle to the X and Y axes experiences combined effects of both row and column beamforming and its gain pattern is not identical to a linear array with elements $\frac{\lambda}{2}$ apart from each other. Considering that the distance between two points along the azimuth angle is $d\sin(\theta)$ as shown in Figure 3.8, the distance between the consecutive microphones is guaranteed to be less than the required distance. Therefore, although the gain pattern is not circularly symmetric extension of the linear array, the planar array will function without ambiguity with varying azimuth angles. More detailed analysis of a planar array's gain pattern will be accomplished using simulations in the upcoming chapters.

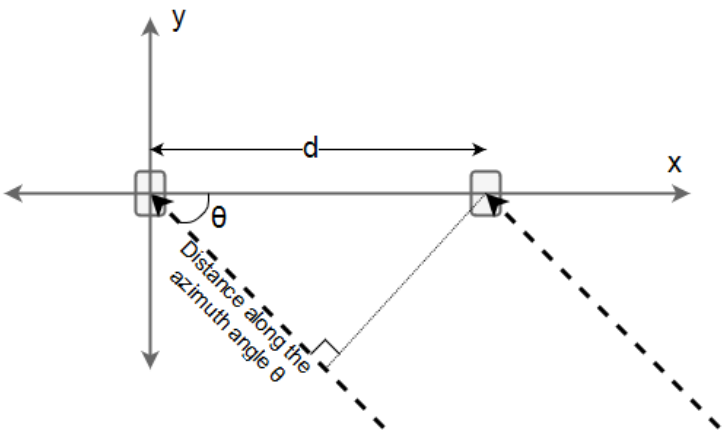


Figure 3.8: The distance between microphones along the azimuth angle.

3.8 Transmitter design parameters

The shape of transmitted signals also effects the capabilities and limitations of an ultrasound system. The transmitted signal used in this research is a pulsed sinusoid, which consists of short bursts of a narrow-band sinusoidal signal as shown in Figure 3.9. While it is possible to improve the range resolution using techniques like pulse compression, such signal requires a design of broadband beamformer that is not within the scope of this research.

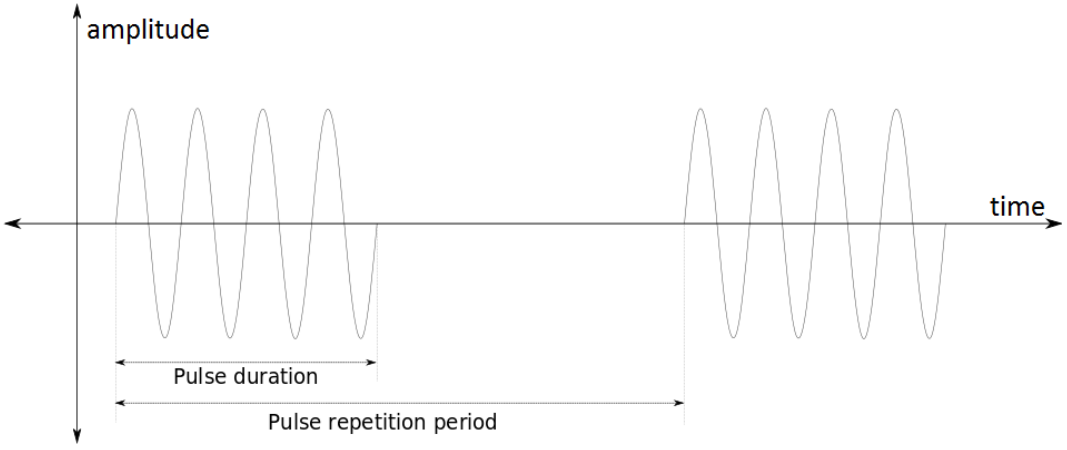


Figure 3.9: Pulsed sinusoid

The frequency of sinusoidal signal depends mainly on the choice of the microphones used for receiving the signal. As discussed previously, the ideal distance between elements in an array is $\frac{\lambda}{2}$. While denser microphone array, and thus smaller distance between elements, is preferred; the distance is physically limited by the size of the microphones. Therefore, given the microphones have sufficient bandwidth to capture the transmitted frequency, it is preferable

to determine the frequency of the sinusoidal signal based on the density of the microphone array achievable using the chosen microphones.

There are two main parameters that can be manipulated when using a pulsed sinusoid signal: pulse repetition frequency (PRF), and pulse duration. PRF, the inverse of pulse repetition period (PRP), determines the range over which the sonar can measure distance to the target object without any ambiguity. If the PRF is too high, the reflection from a far away object could arrive after the transmission of the consecutive pulse. As the returned pulse would be identical to the new pulse reflected from a closer target, the distance to the target becomes ambiguous. Therefore, ultrasound based on a pulsed sinusoid has a hard limit on the range within which it can function correctly. On the other hand, a higher PRF allows faster refresh rate of the sonar, which might be necessary to track a fast moving object. Therefore a balance between the range and the refresh rate must be obtained through the correct choice of the PRF.

The second important parameter, the pulse duration of the transmitter, determines the range resolution of a sonar. Longer pulses allow more energy to be transmitted, hence increasing the SNR (Signal to Noise Ratio) and therefore allow better detection, especially of distant objects. However, a long pulse duration may cause an overlap between reflections from two closely separated objects, therefore limiting the range resolution. Also, while the pulse is being transmitted, the receiver will be saturated by the transmitted signal which leads to a a dead-zone for the duration of the transmission. Therefore the pulse duration must be chosen to enable sufficient SNR, while being as short as possible to maximise the range resolution and minimise the dead-zone.

3.9 Conclusion

In this chapter, the necessity of designing a highly directional and steerable aperture for producing an image has been explained and array beamforming is suggested a solution to effectively create such an aperture.

Based on the propagation model of a far-field source, mathematical models for two beamforming techniques, phase shift beamforming and MVDR beamforming, have been shown. The models were first discussed in a 2D space using linear arrays, and they were expanded to planar arrays by incorporating the azimuth angles into the calculation of the steering vectors. The model for phased array beamforming was also used to predict the performance of beamforming array and its limitations. In Chapter 5, the mathematical models are simulated in software to test the performance of MVDR beamforming compared with phased array beamforming.

On the transmitter's side, the effects of transmitted signal's PRF and pulse duration are discussed. While the exact choice of the parameters will be effected by available hardware, the understanding of their consequences allows

us to make more suitable choices. The discussions in this chapter forms the basis of ultrasound microphone array design that will follow.

Chapter 4

Hardware design

There have been many attempts to use ultrasound for gesture recognition by both large companies and hobbyists. Basic hardware for sonar can easily be built using an off-the-shelf ultrasound range detector or even with a set of consumer grade speakers and microphones. However, using the off-the-shelf products that were not designed specifically for ultrasound imaging will likely limit the functionality and accuracy of the outcome. Hence a specially designed hardware suitable to perform sonar imaging for gesture recognition will be designed and built for the purpose of this research.

In this study, a 2 dimensional beamforming microphone array will be used to create a digitally steerable directional receiver for ultrasound imaging. The hardware consists of three parts: microphone array, ultrasound transducer, and data acquisition. Once the ultrasound data is captured, the remaining beamforming steps are implemented in software. In this chapter, the choice of the hardware components for this project as well as its technical properties are explained.

4.1 Microphone array

The microphones are the most important hardware components of this research, and therefore they were chosen first. Frequency range and physical dimensions are the two factors considered when choosing the microphones for a beamforming array. First of all, the microphone needs to be capable of capturing ultrasound signals. As most microphones are designed to work in the audible range of audio frequencies, this requirement already narrows down the options. The size of the microphone is also an important factor in an array design, as it physically limits the minimum distance between the microphone elements. As discussed in Chapter 3, the ideal distance between the elements is half of the wavelength. Therefore a microphone with a small footprint is preferred, as it allows more flexible placement of the element on a PCB.

Micro-electro-mechanical system (MEMS) based microphones are readily

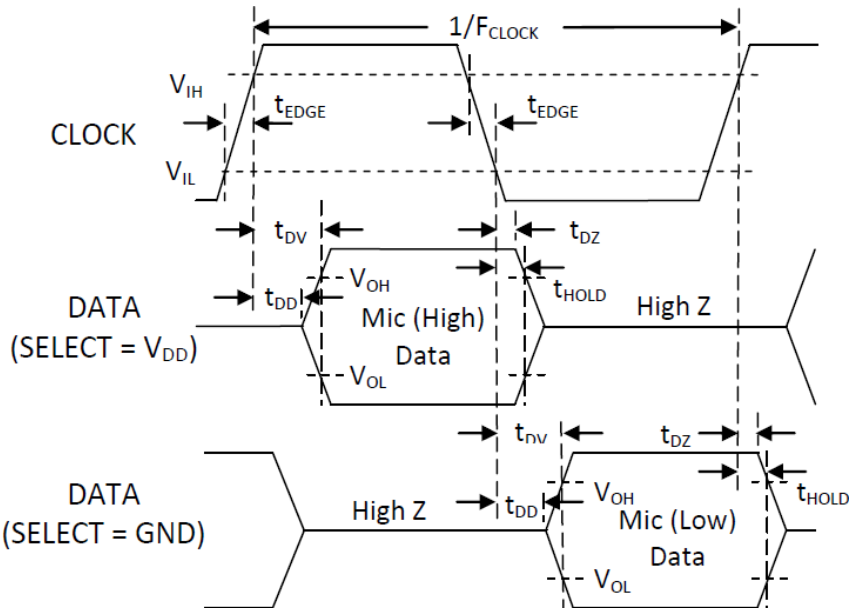


Figure 4.1: Timing diagram of the SPH0641LU4H-1, reproduced from the datasheet (Knowles Electronics, 2014).

available for ultrasound sensing, have small footprints, and are low in cost. This makes them well-suited for building a microphone array. Furthermore, MEMS microphones are available with built-in analogue to digital converter (ADC), thereby eliminating the need for a separate ADC and associated pre-amplification circuits. For a beamforming array, which must acquire a multitude of high frequency signals in parallel, the high tolerance against interference offered by a digital signal is particularly attractive. The SPH0641LU4H-1 manufactured by Knowls is a MEMS microphone capable of capturing ultrasound that has a small footprint, with an integrated ADC. It satisfies all of the desired properties for use in our microphone array and therefore it was selected for use in this project.

The output of the SPH0641LU4H-1 microphone is digitised by a delta sigma modulator and coded as a pulse density modulated (PDM) digital signal. This PDM signal is synchronised to the clock frequency, which can range from 3.072 MHz to 4.8 MHz for ultrasound. For PDM, this bit rate is not directly related to the Nyquist frequency. Instead, the SiSonic design guide recognises the upper frequency limit of a typical ultrasound MEMS microphone as 80 kHz, which is also the upper boundary of the frequency response supplied in the datasheet of the SPH0641LU4H-1. Therefore the design assumes the microphone has a usable frequency response up to 80 kHz.

The SPH0641LU4H-1 has a pin that selects the data output timing to correspond to either the high or the low period of the clock, as shown in the Figure 4.1. This allows two microphones to share a single data line without additional hardware. While this feature is designed to be useful in a stereo

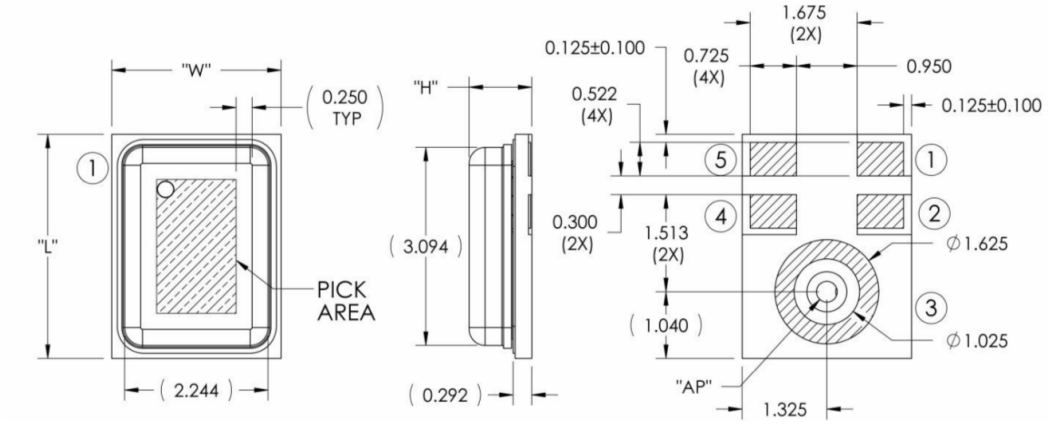


Figure 4.2: Mechanical drawing of SPH0641LU4H-1 (Knowles Electronics, 2014).

microphone configuration, we will make use of it to reduce the number of digital input pins needed while doubling the sampling rate.

According to the datasheet, the SPH0641LU4H-1 is an omnidirectional microphone with no directivity. The gain of the microphone was confirmed to be omnidirectional as claimed. However, the phase of the incoming sound was affected by the direction of the acoustic hole - marked as AP in Figure 4.2 - when tested. This effect was only observed once the microphone was mounted, when the sound had to travel around the PCB to reach the acoustic hole. Also, the effect was not observable for low frequency signals with a longer wavelength. While a phase shift is unimportant for sound recording, it has a devastating effect on beamforming as it becomes impossible to relate the phase shift to the direction of the source or vice-versa. Once this problem was identified, it was easily solved by reversing the orientation of the circuit board. However, it highlights the importance of the orientation of the acoustic hole when choosing a MEMS microphone.

Previous research has shown that an array of 7 microphones was suited for imaging at a resolution appropriate for gesture recognition (van Willigen *et al.*, 2014). In this research, a B-mode image, which is a two-dimensional image showing distance against horizontal azimuth, was successfully captured using a linear equispaced array of 7 MEMS microphones with a 40 kHz signal. From the B-mode image, the two hands of the participant were clearly distinguishable. We aimed to improve on this result by using a 3 dimensional image of a scene with comparable resolution. Therefore the microphone array must be a 2 dimensional rectangular grid, with at least 7 microphone elements per side. As the microphones are operating in pairs, an 8×8 square grid microphone array is chosen as the geometry of our beamforming array. Using equation Equation 3.18, 8 elements linear array can achieve angular resolution of 6.36° and the same angular resolution applies to the planar array for all azimuth angles. This angular resolution is not sufficient to identify fine details within a

hand such as individual fingers, but it should be sufficient to distinguish hands as shown in van Willigen's research. The angular resolution is to be improved through the use of MVDR beamforming.

4.2 Ultrasound transmitter

The microphone chosen in Section 4.1 has the length of 3.094 mm, which allowed the spacing between microphone elements to be 4.25 mm from each other. The ideal wavelength of the transmitted signal, according to the previous discussions, is therefore 8.5 mm. Assuming speed of sound in air to be 340 m/s, this distance makes the ideal transmitter frequency for beamforming to be 40 kHz. The 40 kHz frequency is about half of the microphones' bandwidth on specification, hence the signal can be captured without nearing the microphones' capability in bandwith. Conveniently, 40 kHz ultrasound is also often used for ultrasound range finder and therefore the hardware is easy to find. MA40S4S is one of such transducers that has operating frequency of 40 kHz, and therefore was chosen as the transducer.

The hardware realisation of a pulsed sinusoid generator consists of two parts. The first part is an oscillator that generates a constant 40 kHz sinusoidal signal. This is achieved using a quad op-amp circuit as shown in Figure 4.3 (LearningaboutElectronics, 2016). The second part generates pulses of desired pulse duration and PRF. The pulse generator consists of a monostable IC that is activated by the signal from the clock. The generated pulse is then used to gate the sinusoidal signal from the quad op-amp circuit, using two transistors as an analogue gate. The final design of the pulsed sinusoid generator is shown in the Figure 4.4. Having the pulse synchronised to the main clock allows accurate repetition rate in number of samples, making it easier to interpret the data after acquisition.

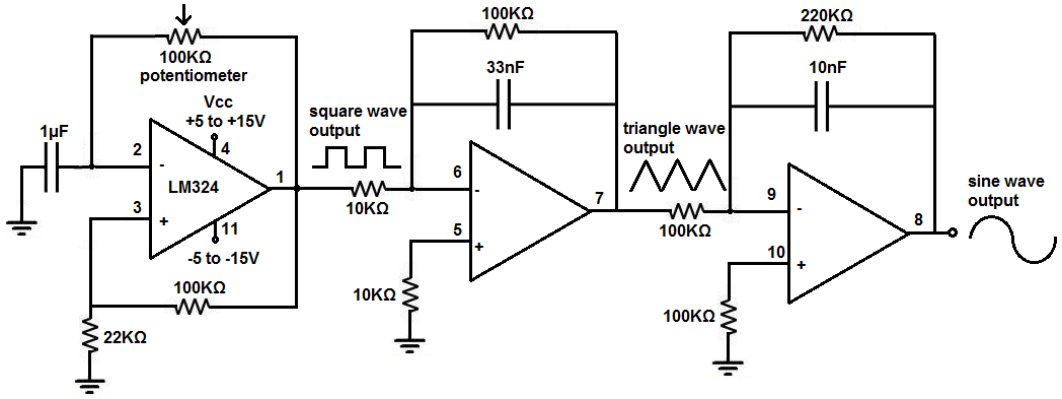


Figure 4.3: Quad op-amp sinusoid generator (LearningaboutElectronics, 2016).

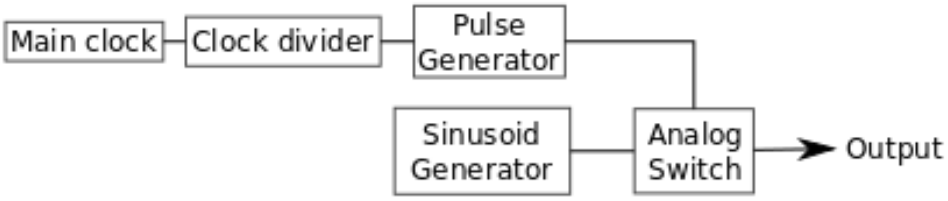


Figure 4.4: Block diagram of the pulsed sinusoid generator

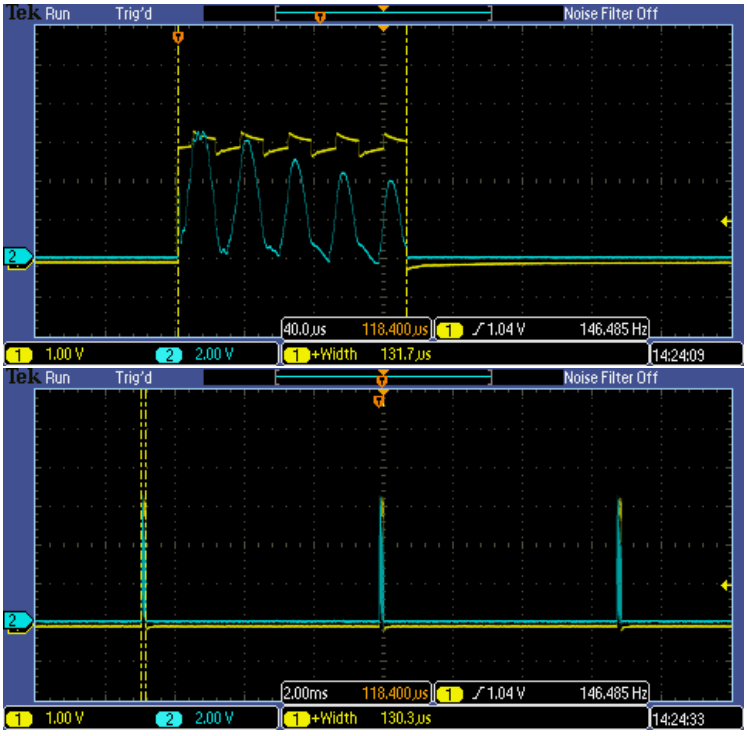


Figure 4.5: Output of pulsed sinusoid generator.

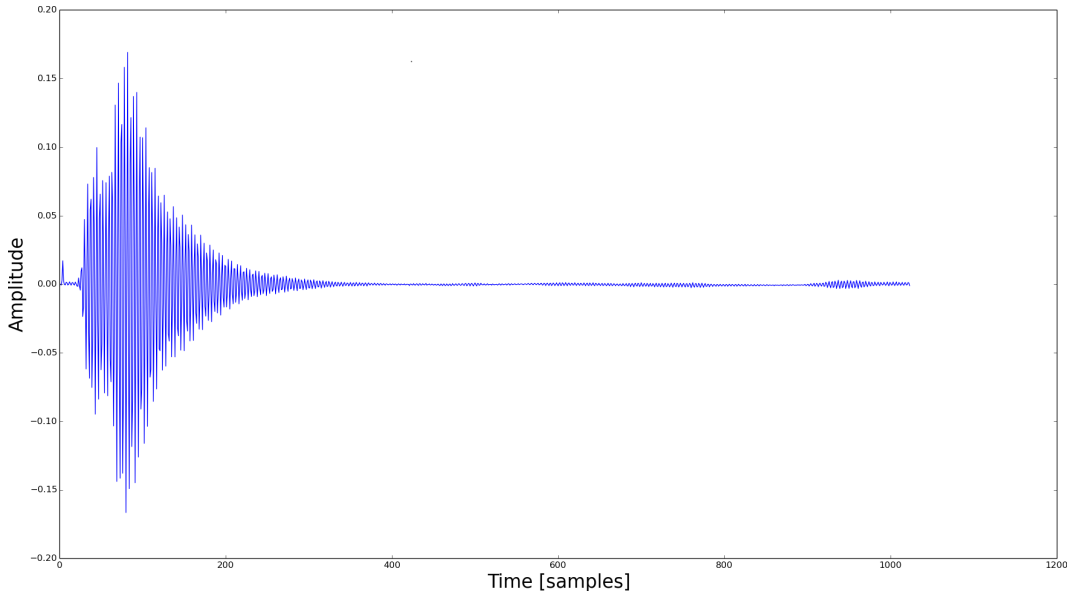


Figure 4.6: Actual reading of the pulsed sinusoid.

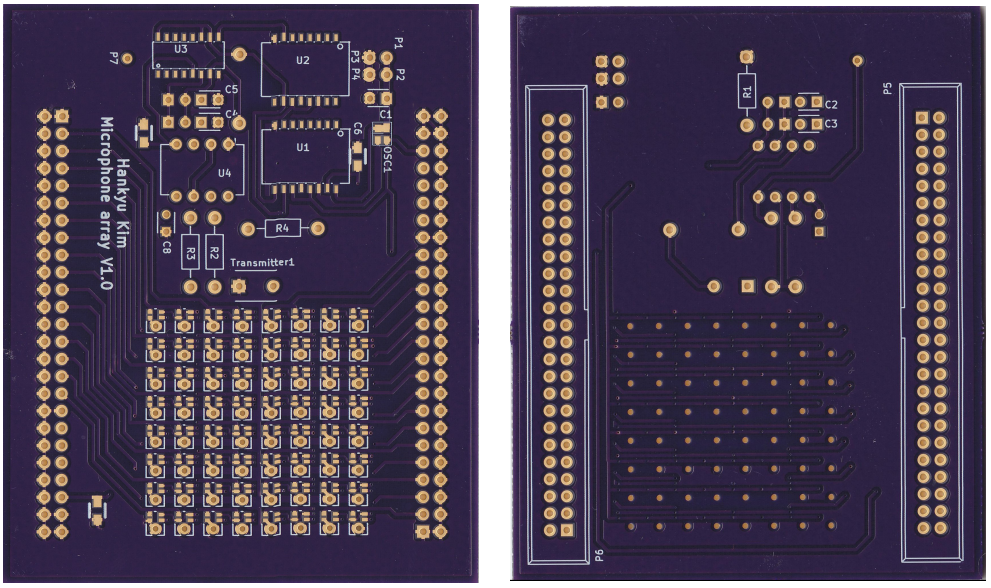
The main clock is used to control the sampling of the microphones and is a 9.6 MHz square signal. A clock divider lowers this frequency to 146.5 Hz using two 8-bit counters. The slower clock is used to activate the pulse generator, so that the pulsed sinusoid will have PRF of 146.5 Hz or once every 2^{16} clock cycles. The monostable multivibrator is set up to create a pulse of 120 μs at each lowered clock signal. Together, the circuit generates a 40 kHz sinusoidal pulse with the pulse repetition rate of 146.5 Hz and pulse duration of 130 μs as shown in Figure 4.5. The top and the bottom pane of Figure 4.5 show the same signal at different time scales. Yellow signal in Figure 4.5 is the control pulse, and the blue signal shows the output. The signal was adjusted to have a pulse duration with start and end times at zero crossing point to reduce the transients included in the sinusoidal pulse. Clearly, the signal generated using the quad op-amp IC is not an ideal sinusoid. Multiple pulsed sinusoid generators were built and tested in order to improve the signal. However, due to the poor transient response of the transducer, the actual ultrasound transmitted degrades further from the design.

Figure 4.6 shows a record of the pulsed sinusoid, sampled at 150 kHz. From the figure, the transmitted signal is at least 150 samples or 1 ms long, with a considerable portion of transient noise. The final specification of the pulsed sinusoid is shown in the Table 4.1. While the measured parameters are not ideal, they are sufficiently good for the purpose of our design. Since the transducer was shown to be the limiting factor in pulsed sinusoid transmitter, the sinusoid generator was not improved further.

Based on the above design for the transmitter and the microphone array, a circuit board was designed and printed. The final design of the board, before soldering in the components, is shown in Figure 4.7.

Parameter	Value
Pulse repetition frequency	146.485 Hz
Unambiguous range	1.16 m
Pulse duration	1 ms
Dead zone	0.17 m
Signal frequency	40 kHz

Table 4.1: Parameters of the pulsed sinusoid signal.



(a) Front (b) Back

Figure 4.7: PCB design for the ultrasound array.

4.3 Data acquisition

Acquiring ultrasound data in parallel from 64 microphones is not a trivial task, even using a modern hardware. The SPH0641LU4H-1 microphone digitises analogue audio signal into a digital PDM signal. Since most digital signal processing algorithms cannot be applied directly to a PDM data, the data must first be converted to pulse code modulated (PCM) signal before further processing. Typically, a dedicated DSP would be used to convert PDM to PCM in real time. However, simultaneously demodulating an array of PDM data is not a common task and an integrated circuit that can demodulate multiple PDM signals could not be found. One solution to this problem is to use an FPGA. A sufficiently large FPGA will be capable of reading and demodulating the PDM signals from the 64 microphones, and also do much of the front-end work; thereby reducing the load of data for the software. Alternatively, the raw digital data can be stored on a PC and demodulated during software post processing. While the software approach is much slower and requires storage

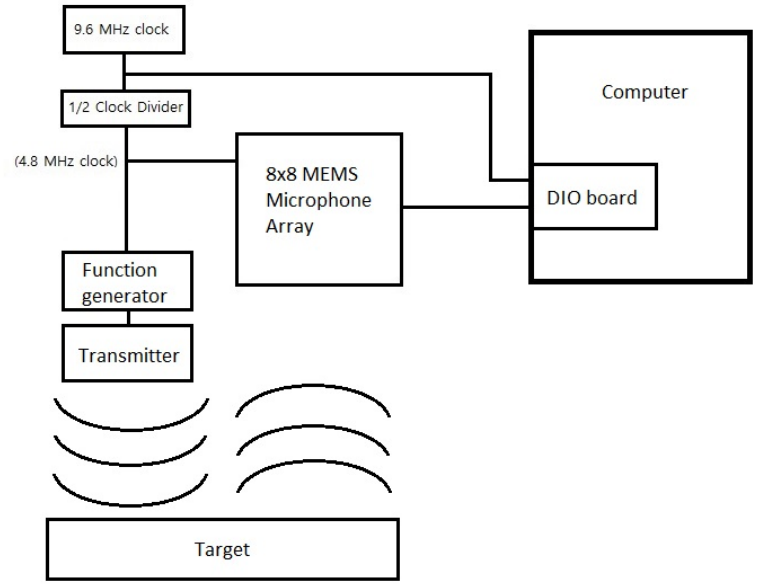


Figure 4.8: System block diagram

for the raw data, it is easier to implement and it is more flexible during system development. Since the focus of this project is on simplified hardware, software demodulation of the PDM was chosen for ease of development even though it means real-time processing may not be possible.

As explained in Section 4.1, the data acquisition (DAQ) hardware needs to have 32 digital inputs, each sampled at 9.6 MHz. After testing a number of DAQ devices, the Adlink-7350 was selected. This device is a high-speed digital input/output board, which has 32 channels. It can read digital data at a rate up to 100 MHz when synchronised with an external clock. The board is attached to a computer workstation using an PCI-E interface and is accompanied by a basic software API for simple read and write functions. Firmware was written in C to read the data and store it in a file. The data read is triggered by the pulse gating the pulsed sinusoid. This allows the data input to be synchronised to the beginning of a transmission, hence the images can be aligned to the transmission correctly. Figure 4.8 shows overall set up of the system.

4.4 PDM

Pulse density modulation (PDM), also commonly known as delta-sigma or sigma-delta modulation, is the modulation used for digital encoding of analogue audio by many MEMS microphones including SPH0641LU4H-1. PDM signals must be demodulated to obtain the conventional pulse code modulation (PCM) coding required for further processing or reproduction as sound.

PDM is therefore not normally used as the main method of storing the audio data. However, PDM has features that make it much easier to implement on a silicon chip, including: relaxed anti-aliasing filter requirements at the pre-amplifier due to its oversampling property, and a purely digital design. The simplicity of the silicon implementation has made it a popular modulation scheme for small ADC systems such as the ones needed in MEMS microphones. Although delta-sigma ADC is an established technology, the idea can be obscure to those who are unfamiliar with it and therefore modulation and demodulation of PDM will be described in the following sections.

4.4.1 Modulation

In order to understand PDM, it is helpful to compare it with conventional PCM. PCM represents an analogue signal by periodically sampling and storing the binary encoded values of each sample. The number of bits used to store each sample is called the bit-depth, and it determines the resolution of the PCM. A PCM audio with the bit-depth of N can represent 2^N quantization levels. The most distinguishing feature of PDM is that it allocates only one bit per sample to represent the data. Single bit resolution sampling greatly simplifies the hardware of the PDM modulator and allows the design to be completely digital. However, low bit-depth results in an extremely large quantisation error. The process of PDM can be understood as a process of reducing the large quantisation noise caused by the low bit resolution.

To see how the single bit representation effects the quantization noise, the effect of bit-depth on the signal to quantization noise ratio (SQNR) of PCM is calculated. For evenly distributed quantisation levels, the sampling error e[n] of N bit word PCM ADC is in the range of $-\frac{Q}{2} \leq e[n] \leq \frac{Q}{2}$ where Q is the quantization step size that is $\frac{1}{2^{N-1}}$ for a normalised input. Assuming the error is distributed uniformly over the range Q, the probability density function of the error is given as Equation 4.1.

$$P(e) = \begin{cases} \frac{1}{Q} & \text{if } |e| \leq \frac{Q}{2} \\ 0 & \text{otherwise} \end{cases} \tag{4.1}$$

The signal power of the quantisation error is given by the second moment of its distribution, and hence it can be derived for given number of bits as shown in Equation 4.2. Since the power of a normalised sinusoid is equal to $\frac{1}{2}$, the signal to quantization noise ratio for a full-scale sinusoid is the ratio of $\frac{1}{2}$ to the noise power, as shown in Equation 4.3. The derived equation in decibels is easy to apply and simple to understand, therefore it is used as a reference in many ADC performance analyses. The equation shows that the SQNR of a 1 bit ADC can be at most 8.78 dB, which is far too low when compared with the typical PCM recording in 16 bit resolution that provides a 98.08 dB SQNR or 24 bit resolution giving a 146.24 dB SQNR.

$$\begin{aligned}\sigma_e^2 &= E(e^2) \\ &= \frac{1}{Q}\int_{-\frac{Q}{2}}^{\frac{Q}{2}} e^2 de \\ &= \frac{Q^2}{12} \\ \sigma_e^2 &= \frac{1}{3\times 2^{2N}}\end{aligned} \tag{4.2}$$

$$\begin{aligned}SQNR &= \frac{3\cdot 2^{2N}}{2} \\ SQNR_{dB} &= 10\log(\frac{3\times 2^{2N}}{2}) \\ &= 10\log(\frac{3}{2}) + 10N\log(2^2) \\ SQNR_{dB} &\approx 1.76 + 6.02\times N\end{aligned} \tag{4.3}$$

The quantisation noise of the PDM signal is reduced in two ways. Firstly the signal is oversampled by sampling it at at a rate much higher than the Nyquist sampling rate. As the quantization error is assumed to be random and uniformly distributed, the error signal power σ_e^2 is distributed evenly in the frequency domain, and has a uniform constant power distribution of $\frac{\sigma_e^2}{f_s}$. Hence oversampling distributes the quantization noise over a larger frequency range, effectively reducing the quantization noise within the band of interest. Integrating over the signal bandwidth B gives in-band noise power for an oversampled signal as $\sigma_{OS}^2 = \frac{\sigma_e^2}{f_s}B$. This shows that increasing the sampling rate f_s in effect reduces the in-band noise by the factor of $\frac{f_s}{B}$ which is known as the oversampling ratio (OSR). Calculating the SQNR for the oversampled signal results in Equation 4.4.

$$\begin{aligned}SQNR &= \frac{3\cdot 2^{2N}}{2}\cdot OSR \\ SQNR_{dB} &= 10\log(\frac{3}{2}\cdot 2^{2N}\cdot OSR) \\ SQNR_{dB} &\approx 1.76 + 6.02\left(N + \frac{\log_2(OSR)}{2}\right)\end{aligned} \tag{4.4}$$

Equation 4.4 shows that, each time the sampling frequency is doubled, the effective bit-depth is increased by half a bit. Note that the total noise power over the entire band is unchanged, and that the out-of-band noise must be

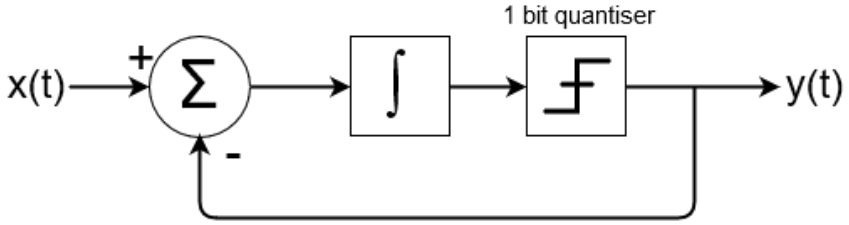


Figure 4.9: First order delta-sigma modulator.

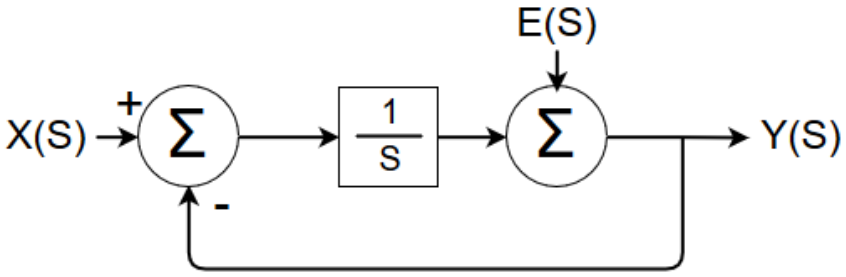


Figure 4.10: Laplace transformed delta-sigma modulator.

filtered out at the demodulation stage in order to achieve this improvement in SQNR.

The in-band quantization noise is further reduced by means of the technique of noise shaping. Noise shaping is often used to improve the perceived signal to noise ratio by shifting the noise into a frequency band where it is less perceptually significant. In our case the noise is filtered by a digital high pass filter, which attenuates the noise in the baseband of an oversampled signal. In a delta-sigma ($\Delta\Sigma$) ADC this can be represented as shown in Figure 4.9. To understand the operation of the first order delta-sigma modulator, Figure 4.10 shows the Laplace transform of the system in Figure 4.9. X(S), Y(S) and E(S) are the Laplace transforms of the input, modulated output and quantization noise respectively. The 1 bit quantiser is modelled as an additive error E(S), which is the Laplace transform of a uniformly distributed random error e(t) between -0.5 and +0.5 as for PCM. By considering the transfer function from the inputs X(S) and E(S) to the output Y(s), we can characterise the effect of the modulator. Because the system is linear, these transfer functions can be found separately by considering one input at a time, while making the other input zero. The result is shown in Equation 4.5 for X(s) and Equation 4.6 for

E(s) (Park and Motorola, 1990).

$$\frac{Y(S)}{X(S)} = \frac{1}{S+1} \tag{4.5}$$

$$\frac{Y(S)}{E(S)} = \frac{S}{S+1} \tag{4.6}$$

Equation 4.5 and Equation 4.6 are the transfer functions of a first order low pass and high pass filter respectively. Therefore, this frequency analysis shows that the delta-sigma modulator effectively low pass filters the input signal X(s) and high pass filters the quantization noise E(s).

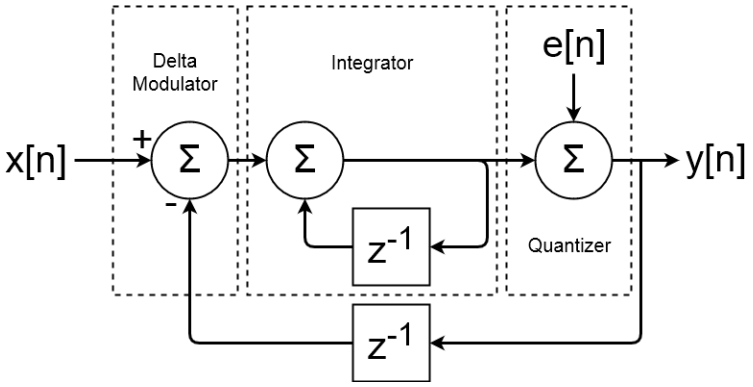


Figure 4.11: Discrete time delta-sigma modulator.

The discrete time implementation of the delta-sigma modulator is shown in Figure 4.11. Evaluating Figure 4.11 shows that the output y[n] of the digital system is the sum of input signal x[n] and high pass filtered noise as shown in Equation 4.7.

$$Y(z) = X(z) + (1 - z^{-1})E(z) \tag{4.7}$$

The analysis of the transfer functions for the error in discrete time shows that the quantisation noise $e[n]$ is filtered with a first order high pass filter. The PDM modulator described in Equation 4.7 was implemented using Python and the source code can be found in Appendix B. Figure 4.12 shows the FFT of a pulse density modulated sinusoidal signal, generated using the code in Appendix B. In the figure, two closely-spaced sinusoidal components, corresponding to the positive and negative frequency components of the single real sinusoid, are visible, as well as superimposed high-pass filtered white noise. The figure clearly shows the first order high pass filtering of the noise, as predicted from the analysis. The baseband of the output maintains a low noise level and therefore allows the signal to be easily distinguished.

Based on the discrete time transform function in Equation 4.7, it is possible to calculate the SQNR of a PDM signal (Mladenov *et al.*, 2011). The power

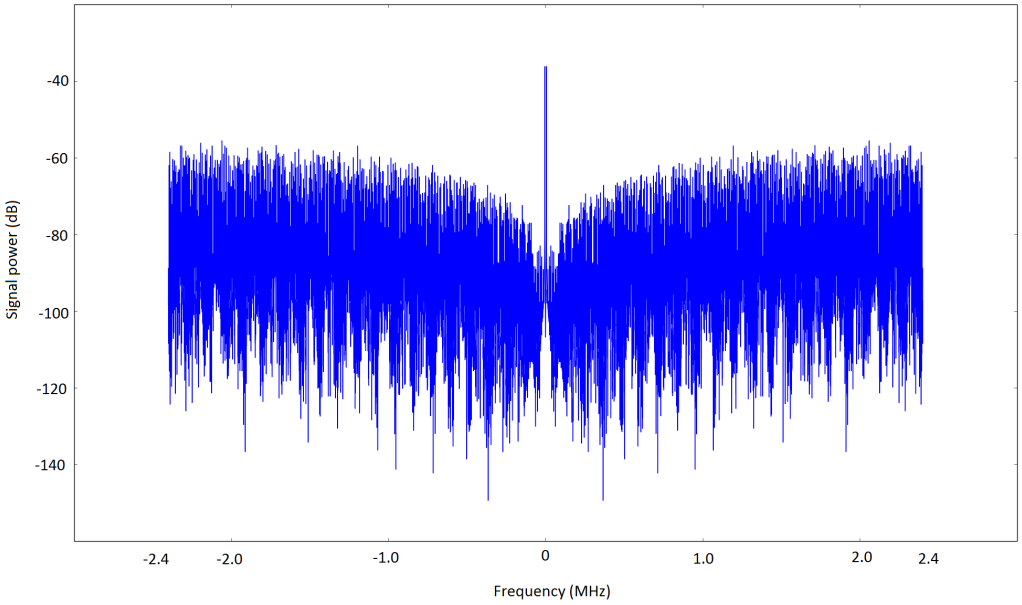


Figure 4.12: PDM modulated sine wave.

spectral distribution of the noise in the oversampled signal is a constant $\frac{\sigma_e^2}{f_s}$, and it is filtered by a transfer function $(1 - z^{-1})$ as shown in Equation 4.7, which we will call noise transfer function. To calculate the noise power at the output, the input noise power is multiplied by the transfer function squared and integrated over the bandwidth B of the signal as shown in Equation 4.8. Following the steps in Equation 4.9, the magnitude of noise transfer function is $2 - 2\cos(\frac{2\pi f}{f_s})$. Substituting and integrating the two equations results in Equation 4.10.

$$\sigma_n^2 = \int_{-\frac{B}{2}}^{\frac{B}{2}} S_e^2(f)\ |NTF(f)^2|\ df \tag{4.8}$$

$$\begin{aligned} |NTF(f)^2| &= |1 - z^{-1}|^2 \\ &= (1 - e^{j\omega})(1 - e^{-j\omega}) \\ &= 1 - e^{j\omega} - e^{-j\omega} + 1 \\ |NTF(f)^2| &= 2 - 2\cos(\omega) \end{aligned} \tag{4.9}$$

$$
\begin{aligned}
\sigma_n^2 &= \int_{-\frac{B}{2}}^{\frac{B}{2}} \frac{\sigma_e^2}{f_s}|1 - z^{-1}|^2 \, df \\
&= \frac{\sigma_e^2}{f_s} \int_{-\frac{B}{2}}^{\frac{B}{2}} 2 - 2\cos\left(\frac{2\pi f}{f_s}\right) \, df \\
&= \frac{\sigma_e^2}{f_s}\left[2B - \frac{2f_s}{\pi}\sin\left(\frac{\pi B}{f_s}\right)\right] \\
&= \sigma_e^2 \left[\frac{2B}{f_s} - \frac{2}{\pi}\sin\left(\frac{\pi B}{f_s}\right)\right] \\
\sigma_n^2 &= \sigma_e^2 \left[\frac{2}{OSR} - \frac{2}{\pi}\sin\left(\frac{\pi}{OSR}\right)\right] \qquad (4.10)
\end{aligned}
$$

A typical oversampling ratio of a PDM microphone is 64. Substituting this value the estimated noise power of single bit sampling before noise shaping σ_e^2 into Equation 4.10, the estimated quantization noise in the baseband of PDM signal is $\sigma_n^2 = 1.04569 \times 10^{-6}$. Assuming a normalised sinusoid which has signal power of $\frac{1}{2}$, the SQNR of PDM with an OSR of 64 is 56.80 dB. This corresponds approximately to a 9 bit resolution PCM signal. While it is almost 50 dB improvement over the SQNR of 1 bit PCM, most audio ADC offer 16 bits resolution. Therefore the SQNR of typical PCM is far superior to that of PDM even with 64 times OSR.

However, delta-sigma modulators are often implemented in multiple stages, performing higher-order high pass filtering on the noise for improved suppression in the baseband. Naturally, better suppression of noise in the baseband incurs a reduction in the effective quantization noise and therefore an improved SQNR. Hence the SQNR of PDM can theoretically be made as high as needed, by cascading delta-sigma modulators or by increasing the OSR. Therefore the only limit in the SQNR of a PDM signal is in the complexity of the modulator. There may be limits in practical implementation of the delta-sigma modulators and demodulators, such as non-ideal filter responses. However, we are only interested in demodulating existing PDM signals and thus we do not have to be concerned about such extreme scenarios. Without the knowledge of the design used in the specific modulator, it is not possible to model the quantisation noise of a PDM signal correctly. From the outcome of this analysis, it is sufficient to note that the SNR of PDM is measure of SNR in its baseband and the quantisation noise within the baseband can be suppressed as needed; as long as the remaining high frequency noise is adequately removed.

4.4.2 Demodulation

The preceding analysis of PDM showed that the quantisation noise in a PDM signal is high pass filtered during modulation and therefore exists predominantly in the high frequency bands. Therefore, to demodulate a PDM signal, one needs to remove the oversampled high frequency band along with the quantisation noise. This process is called decimation, and it consists of an anti-aliasing filter and subsequent down-sampling. Down-sampling a signal without filtering the out-of-band noise causes the noise to be aliased into the baseband. Therefore, to prevent the quantisation noise from effecting the demodulated signal, it is important to use a good anti-aliasing filter during the decimation. Any low pass filter can be used for anti-aliasing, including a classic finite-impulse-response (FIR) filter. However, for a PDM signal that has high oversampling ratio, the required roll-off ratio is extremely high, which increases the number of taps needed for the FIR decimation filter. Furthermore, the FIR filter has to be applied at a high sampling rate before down-sampling. Having to perform a large number of multiplications to apply the tap weights on a high-sample-rate signal makes the FIR decimation filter an extremely inefficient choice for demodulating a PDM signal. A cascaded-integrator-comb (CIC) filter is much better suited for large decimation ratios in terms of computational efficiency. However, the CIC filter has a non-flat response that is inferior to a FIR filter. Therefore, in this research, a combination of a CIC filter and a FIR filter is used to demodulate the PDM signal.

While there is no standardised decimating filter designated for demodulating a PDM signal, a CIC filter is widely used as it can efficiently perform decimation for highly oversampled signal (Wang and Reiss, 2012). The integrator-comb filter is an alternative implementation of the moving average filter, and a CIC filter is the cascaded version of it. A typical implementation of moving average filter's computational load depends on the averaging block-size. The integrator-comb filter is designed to make its computation independent of the block length, making it efficiently cascadable (Lyons, 2005).

The optimisation of integrator-comb filter is achieved by first accumulating values (integrator) and then subtracting values after a delay (comb). Hence averaging only needs one addition and one subtraction, regardless of the block-size. Also, the filter only requires a delay line of block-size in length, and does not require access to the intermediate signals. Multiplier-less arithmetic and the lack of need for a constantly accessible memory makes it possible to implement a CIC filter extremely efficiently, directly in hardware (e.g. FPGA).

Mathematically, the output and the transfer function of a moving average

filter with a block-size D is shown in Equation 4.11.

$$\begin{aligned} y(n) &= \frac{1}{D}\left(x(n) + x(n-1) + x(n-2) + \cdots + x(n-D+1)\right) \\ &\mathcal{Z}\downarrow \\ Y(z) &= \frac{1}{D}X(z)(1 + z^{-1} + z^{-2} + \cdots + z^{-D+1}) \\ H(z) = \frac{Y(z)}{X(z)} &= \frac{1}{D}\sum_{n=0}^{D-1}(z^{-n}) \end{aligned} \tag{4.11}$$

Using the closed-form expression for the sum of a geometric sequence, the filter can be rewritten as a feedback loop with an accumulator and a subtractor after delay, forming an integrator-comb filter shown in the Equation 4.12.

$$\begin{aligned} H(z) = \frac{Y(z)}{X(z)} &= \frac{1}{D}\left(\frac{1 - z^{-D}}{1 - z^{-1}}\right) \\ Y(z)(1 - z^{-1}) &= \frac{1}{D}X(z)(1 - z^{-D}) \\ Y(z) &= \frac{1}{D}(X(z) - X(z)z^{-D}) + Y(z)z^{-1} \\ &\mathcal{Z}^{-1}\downarrow \\ y(n) &= \frac{1}{D}\left(x(n) - x(n-D)\right) + y(n-1) \end{aligned} \tag{4.12}$$

Because a PDM signal has a bit depth of 1, the filtered signal $y(n)$ in Equation 4.12 will have a value between 0 and 1. However, by ignoring the averaging factor $\frac{1}{D}$, the filtered signal can be represented exactly as an integer, with a bit depth of $\log_2(D)$. While the digital representation is preferable, it has to be ensured that the bit growth does not exceed the size of the data type used to store the value. Without the factor, the expression of a single stage transfer function of a CIC filter becomes Equation 4.13. For an M^{th} order cascaded filter, the transfer function is simply raised to the M^{th} power, as shown in Equation 4.14.

$$\begin{aligned} y(n) &= y(n-1) + x(n) - x(n-D) \\ H(z) &= \frac{1 - z^{-D}}{1 - z^{-1}} \end{aligned} \tag{4.13}$$

$$H(z)^M = \left(\frac{1 - z^{-D}}{1 - z^{-1}}\right)^M \tag{4.14}$$

The frequency response of an integrator-comb filter can be found by setting $z = e^{j2\pi f}$ in Equation 4.13. The result is given in Equation 4.15 and shows that the CIC filter is similar to a sinc filter. It has troughs where normalised frequency is a multiple of $\frac{1}{D}$. The troughs can be used to minimise the noise

it the baseband by choosing the down-sampling ratio R as a multiple of the block-size D.

$$\begin{aligned} H(e^{j2\pi f}) &= \frac{1 - e^{-j2\pi fD}}{1 - e^{-j2\pi f}} \\ H(e^{j2\pi f}) &= \frac{e^{-j\pi fD}(e^{j\pi fD} - e^{-j\pi fD})}{e^{-j\pi f}(e^{j\pi f} - e^{-j\pi f})} \\ H(e^{j2\pi f}) &= \frac{\sin(\pi fD)}{\sin(\pi f)} e^{-j\pi f(D-1)} \end{aligned} \tag{4.15}$$

The transfer function in Equation 4.13 can be split in to two filters: the integrator $\frac{1}{1-z^{-1}}$, and the comb filter $1 - z^{-D}$. If the block-size is chosen to be a multiple of the down-sampling ratio, the down-sampling can be done before the delay on comb-filter, as shown in Figure 4.13. This reduces the length of the required delay line and the number of subtractions by a factor of the down-sampling ratio, making the CIC filter vastly more efficient than other decimation filters. In order for block-size and down-sampling ratios to be multiples of each other, they are normally set to be equal.

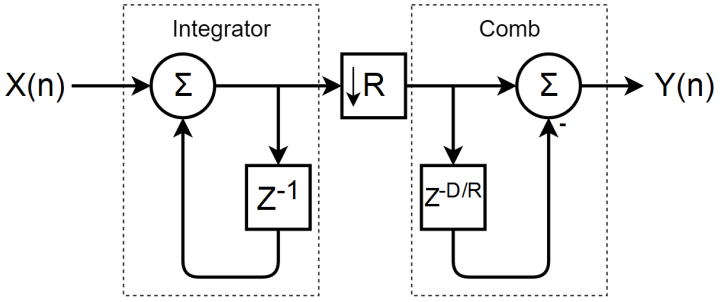


Figure 4.13: Block diagram of an efficient single stage CIC decimating filter.

The noise suppression of a CIC filter improves as the cascading order M increases as shown in the magnitude plot of Equation 4.14 in Figure 4.14 with varying cascading orders. Therefore, with a sufficiently cascaded CIC filter, the quantisation noise of a PDM signal can be suppressed as needed. However, a CIC filter has substantial unwanted attenuation in the pass-band near the stop frequency. This causes higher frequency components within the pass-band of a CIC filter to be attenuated and therefore the recovered signal to be distorted. A CIC filter can be cascaded with a FIR filter to compensate for this attenuation.

The CIC filter can be closely approximated using a sinc function. Therefore, an ideal compensating filter should have a frequency domain response of a band limited inverse sinc function. However, designing a band limited filter with a sharp roll-off will require a high filter order and defeat the purpose of using a CIC filter to improve the efficiency. Hence many compensating

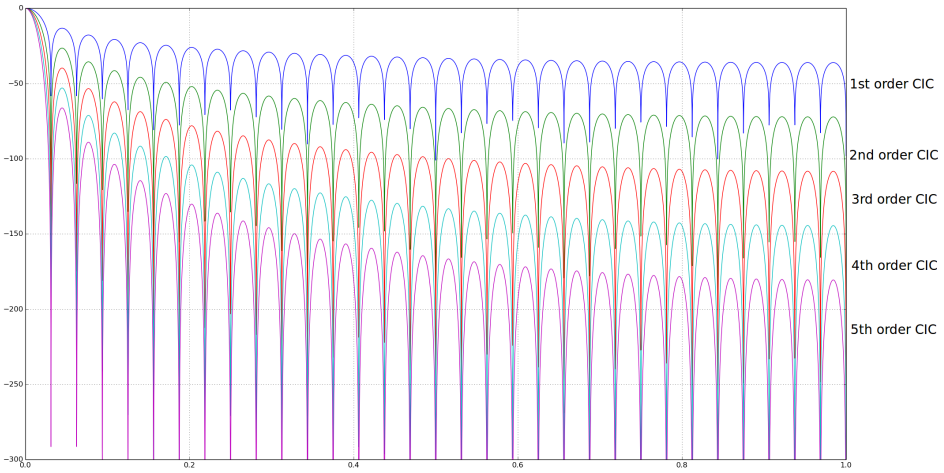


Figure 4.14: Comparison of CIC filter responses with different orders M.

filter designs aim to improve a CIC filter response while using a limited number of taps (Dolecek and Diaz-Carmona, 2011). One method restricts the order of the FIR compensating filter and maximises the combined filter's flatness by ensuring that the derivatives of the frequency response is zero at DC (Fernandez-Vazquez and Dolecek, 2012). The employed filter has linear phase with real symmetric tap weights. It is named a maximally flat CIC compensating filter and it is dependant on the delay length D, decimating ratio R and the order of CIC filter M.

The design begins by assuming a real symmetric L^{th} order FIR filter $P(z)$:

$$P(z) = \sum_{n=0}^{L} a_n z^{-n} \tag{4.16}$$

where $a_n = a_{L-n}$. The symmetry of tap weights ensures that there is no phase distortion and allow us to focus on the magnitude response of the filter. The goal is to design $P(z)$ such that as many derivatives of $|H(z)P(z^D)|$ are zero when $z = 0$, where $H(z)$ is the CIC filter response. Meanwhile the magnitude of the response at $\omega = 0$ should be unity:

$$\begin{aligned} |H(0)P(0)| &= 1 \\ \left| \frac{d^p}{dw^p} H(w)P(Dw) \right|_{w=0} \Bigg| &= 0 \end{aligned} \tag{4.17}$$

Note that the frequency of the compensating filter is raised by the integer factor D, as the resulting compensating filter has to be applied to the decimated signal. The solution requires the use of general Leibniz rule for p^{th} derivative of a product and solving it for up to L^{th} derivative for L^{th} order FIR filter. The

author suggested two types of compensating filters using this rule: a second order FIR filter suited for narrowband signals, and a fourth order FIR filter for compensating wideband signals. Both second and fourth order compensating filters will be tested for the CIC filter in our application.

Following the design rules, the resulting filters are described using two constants A and B that are defined as:

$$A = \frac{1 - D^{-2}}{1 - 2^{-2}}$$
$$B = \frac{1 - (2D)^{-2}}{1 - 2^{-4}}$$

The second order maximally flat CIC compensating filter is a second order FIR filter with 3 taps with weightings as given in the Equation 4.18.

$$P_1(z) = a_0 + a_1 z^{-1} + a_0 z^{-2}$$

$$\text{where} \quad a_0 = -2^{-5} M \cdot A \qquad a_1 = 1 - 2a_0 \tag{4.18}$$

It is possible to improve the compensating filter using a fourth order FIR filter, with 5 taps defined as Equation 4.19.

$$P_2(z) = a_0 + a_1 z^{-1} + a_2 z^{-2} + a_1 z^{-3} + a_0 z^{-4}$$

$$\text{where} \quad a_0 = 2^{-8} M \cdot A(2^{-3} M \cdot A + 1 - 2^{-2} B) \qquad a_1 = -2^2 a_0 - 2^{-5} M \cdot A \qquad a_2 = 1 - 2a_0 - 2a_1 \tag{4.19}$$

Before designing the compensating filter, the parameters of the CIC filter must be finalised. In order to ensure sufficient bandwidth for the 40 kHz signal, the down-sampling ratio R is chosen to be 32. The resulting sampling rate after decimation is 150 kHz. This is almost double the Nyquist rate for a 40 kHz signal and it allows sufficient margin for in-band attenuation by the CIC filter. The block-size is chosen to be equal to the down-sampling ratio, and the filter is cascaded four times. The four cascades provide at least 50 dB suppression of the side-lobes with reasonable computation. Based on the chosen parameters D = 32, R = 32, and M = 4; the compensation filters are designed using Equation 4.18 and Equation 4.19. The frequency response of second and fourth order maximally flat CIC compensating filters were then simulated using the Python code given in Appendix A. The resulting frequency response in Figure 4.15 shows that, while the second order filter overcompensates the signal, the fourth order filter maintains flat response and has less than 1 dB attenuation up to 40 kHz. Therefore the fourth order

compensated CIC filter is used for demodulating PDM signals. Note, although the plot only shows the magnitude of the frequency responses, we can be assured that the phase responses are always linear as the CIC filter and the compensating filter are both FIR filters with symmetric taps. It assures that the filter will be free of phase distortion, and will have a group delay of $\frac{D-1}{2}$ samples, where D is the number of taps (Chen and Parks, 1987). This can also be seen from the phase part of Equation 4.15 for the CIC filter.

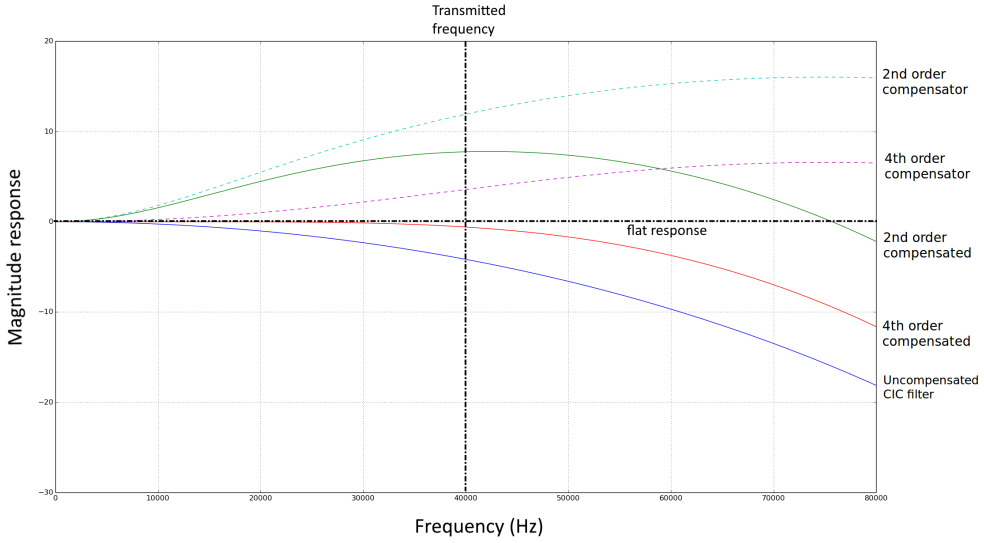


Figure 4.15: Compensated CIC filter responses

4.5 Conclusion

This chapter has described the design of the microphone array and other necessary components of the ultrasound imaging hardware, along with the demodulator needed for captured signals. An 8×8 array of SPH0641LU4H-1 MEMS microphone was chosen for use in the beamforming receiver. The array will provide sufficient beamforming capability needed to capture images for gesture recognition. On the transmitter side, a pulsed sinusoid generator is designed to transmit sinusoidal pulses in synchronisation with the main clock. The echoes of the transmitted signals are captured and encoded as PDM by the microphones and sent to a workstation using Adlink-7350 digital IO board. The data read is triggered by the gating signal of the transmitter, thereby ensuring that each record is synchronised with transmitted pulse. A CIC filter compensated by a fourth order FIR filter was designed to efficiently demodulate the PDM signal, with minimal attenuation at the operating frequency. Next chapter will discuss the software required for beamforming the captured data.

Chapter 5

Beamforming software

The concepts of phased array beamforming and its mathematical model were discussed in Chapter 3. In this chapter, the models developed in Chapter 3 are implemented as software written in Python. It will show how the data from the hardware is processed into an image through a beamforming software. Every model and software used were tested in simulations before being applied to the real data captured from the hardware. Analysing the simulated results reveals some in-depth understanding of the models that were not obvious from the mathematical analysis and provides predictions on what real captured data should look like. Therefore results of the simulations will be analysed and used in the following chapters to validate the real measurements.

5.1 Far field model

The far-field approximation forms the basis of the beamformer design and was tested using a software simulation. The far-field model for the array receiver assumes the received signal to be a planar wave instead of a point source. This assumption makes the phase difference between two receivers proportional to the incident angle of the signal, regardless of the distance from the source. However, the approximation inevitably also adds error to the model.

The transmitted signal is designed to have a dead-zone of 0.044 m as shown in Figure 4.5. However, 0.044 m does not satisfy the far-field condition in Equation 3.19, which requires the distance to the source d to be considerably larger than 0.136 m. In order to compare the difference between the near-field and far-field models, software computing the gain of a linear phased array with 8 microphones, 4.25 mm spacing, and 40 kHz transmitted signal is implemented for both. The parameters correspond to a row of the microphone array hardware designed in Chapter 4. Figure 5.1 shows the array's gain pattern when using the near-field model shown in Equation 3.11 and the far-field model shown in Equation 3.13. The amplitude of the received signal from different incident angles when no steering is applied to the beamformer is shown.

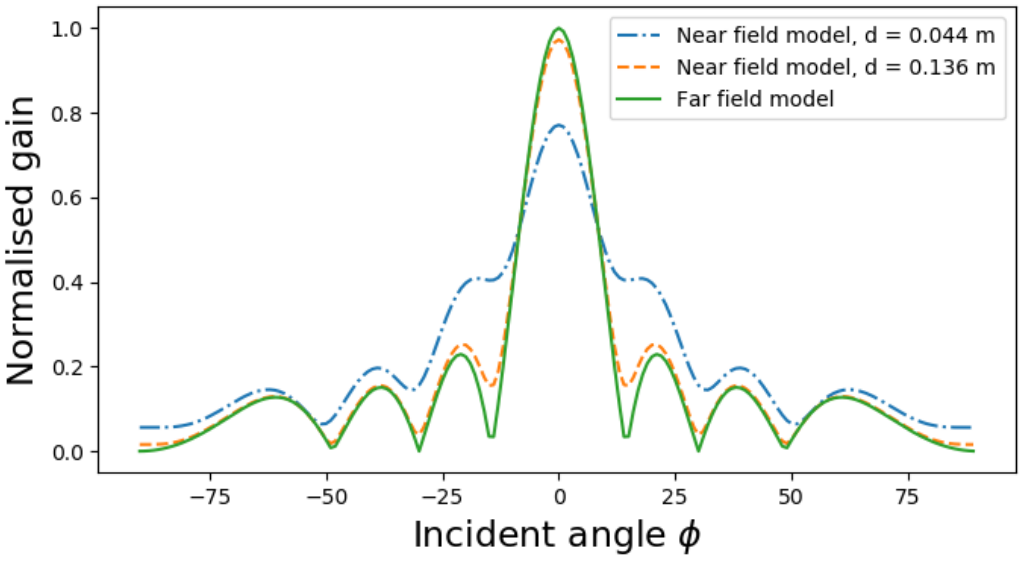


Figure 5.1: Directivity pattern of an 8 element linear array measured using near-field and far-field models.

It is evident that the gain calculated using the near-field model significantly deviates from the far-field model at the distance of 0.044 m away from the receiver.

However, due to the poor transition response of the transmitter, the actual dead-zone acquired from real measurements was found to be 0.17 m. While the far-field approximation fails at a distance of 0.044 m from the receiver, the directivity pattern quickly converges as the distance increases. At the far-field boundary 0.136 m away from the receiver, the directivity pattern of the near-field model is already closely approximated the far-field model. Therefore, at a cost of larger dead-zone, we are able to safely assume that all signals captured outside the dead-zone can be closely approximated using the far-field approximation. When using the hardware, gestures will be performed at least 0.20 m away from the microphone array in order to avoid the dead-zone and also to ensure the validity of the far-field approximation.

The far-field model in Equation 3.13 is again used to produce a 3-dimensional visualisation of the 8×8 microphone array's directivity pattern, this time with varying azimuth angle θ. Figure 5.2, produced by the code in Appendix D, shows the directivity pattern of the array when no delay or phase shift is applied. By simply accumulating the signals captured by the microphone elements, the array acts as a single large aperture with the beam focused at the centre of the array. Note that the directivity pattern is not circularly symmetric due to the rectangular design of the microphone array, and larger side-lobes appear along the x and y axis.

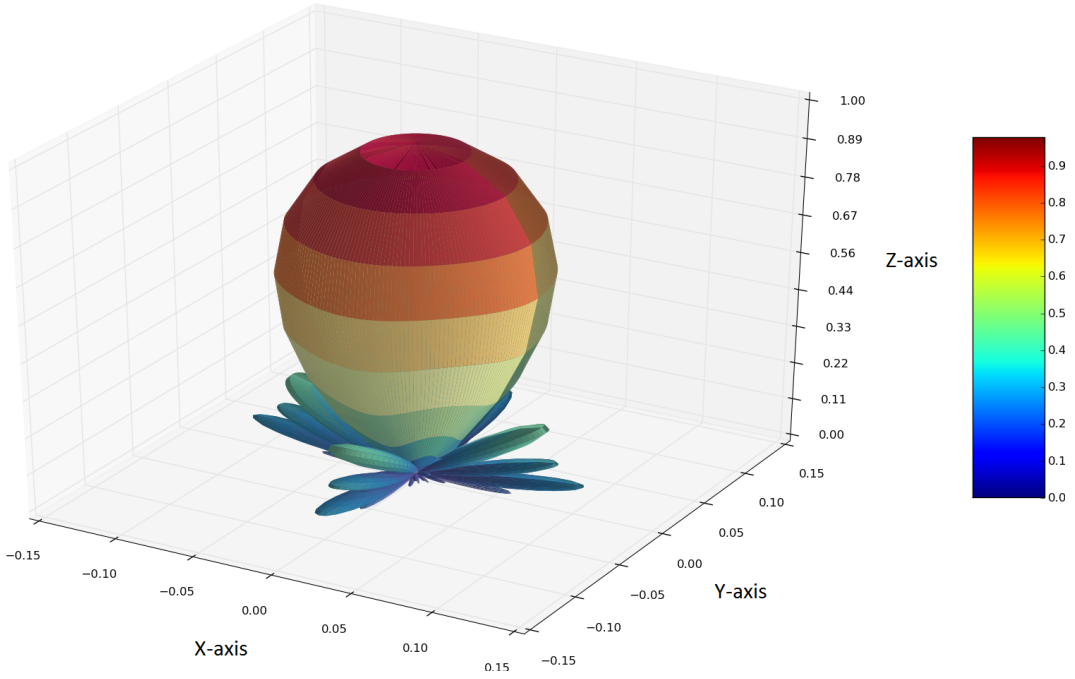


Figure 5.2: 3D directivity pattern of an 8×8 element array.

5.2 PDM modulation and demodulation

The microphones output is encoded using PDM, the modulation and demodulation of which was studied in Section 4.4. In order to test the various filters and beamforming algorithms, it is convenient to be able to generate a PDM signal so that the output can be simulated using known parameters. Python code shown in Appendix B was written to implement first order PDM of a 40 kHz sinusoid signal. The code is an exact implementation of the first order discrete delta-sigma modulator shown in Figure 4.11. To match the file format used by the data acquisition hardware, the code stores the output with each bit repeated 64 times; representing 64 channels of identical interleaved PDM signals. The code in Appendix B generates a continuous 40 kHz sinusoidal, assuming a sampling rate of 4.8 MHz. The generated signal is used to test the demodulator and beamformer algorithms.

The code in Appendix C is based on the demodulator design in Section 4.4. It implements the compensated CIC filter with the filter response shown in Equation 4.15. The code reads an input file in PDM format, deinterleaves the 64 channels, and applies the compensated CIC filter to each channel. The demodulator is tested on the 40 kHz sinusoid generated with the code in Appendix B, and the result is compared to a sampled sinusoid in Figure 5.3. The demodulated 40 kHz sinusoid is delayed and attenuated, as predicted from Equation 4.15, but otherwise identical to the sampled sinusoid that was not demodulated from the PDM.

Both signals in Figure 5.3, sampled at 150 kHz, do not appear as a smooth

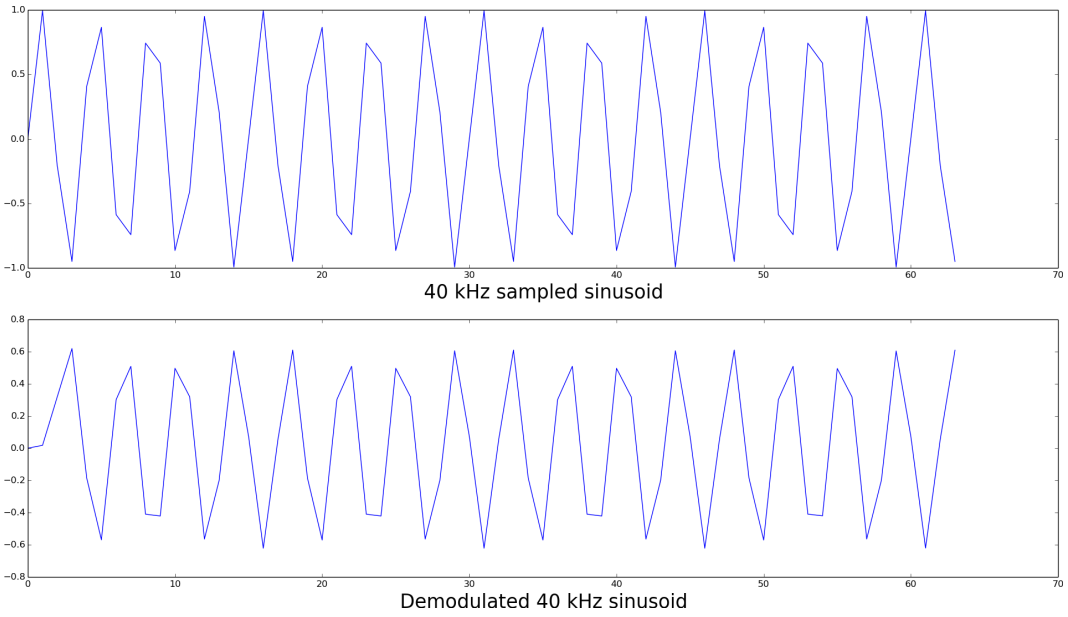


Figure 5.3: A comparison of a 40 kHz sinusoid sampled at 150 kHz (top) and the same signal modulated to and demodulated from PDM (bottom). The bottom signal was sampled at 4.8 MHz before the modulation, but decimated to a sampling rate of 150 kHz during the demodulation.

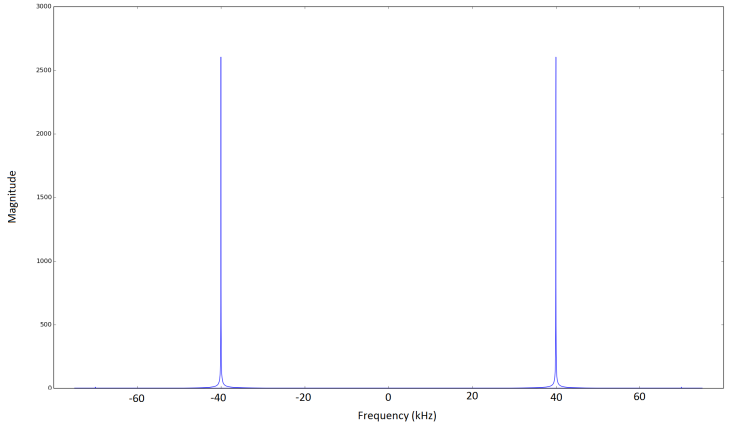


Figure 5.4: Frequency domain representation of the demodulated signal.

sinusoids due the low sampling rate. However, as the sampling rate is chosen to be meet the Nyquist rate, the sampled signal should only consist of the 40 kHz sinusoid regardless of its visual appearance. The frequency domain representation of the demodulated signal shown in Figure 5.4 confirms this. The result assures the accuracy of the pulse density modulator and demodulator implementations; therefore we can use them to demodulate the captured signal with confidence.

5.3 Pulse detection

In order to interpret the signals captured by the beamformer, the returned pulse should be isolated from the beamformed signals. To detect the pulses embedded in the signals, a matched filter is used. A matched filter is implemented by correlating the signal with a template that is identical to the shape of the pulse to be detected. When the template overlaps with the pulse, there will be a high correlation which identifies a match. This provides an efficient means to filter the pulses from a noisy signal. Figure 5.5 shows the matched filter (top), which is a short pulse of sinusoid, being applied to a generated pulsed sinusoid signal with the frequency identical to the template (middle). The pulse highlighted with red box is becomes visible after filtering (bottom).

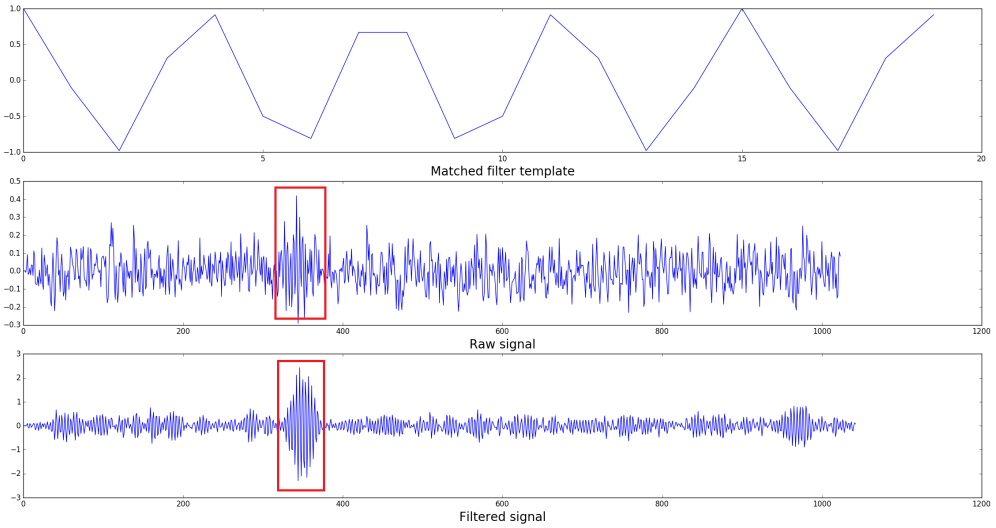
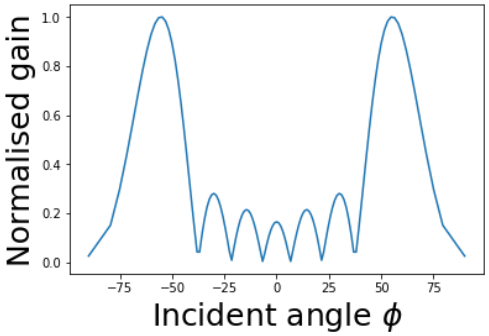


Figure 5.5: Matched filter applied to a generated pulse with artificially added noise.

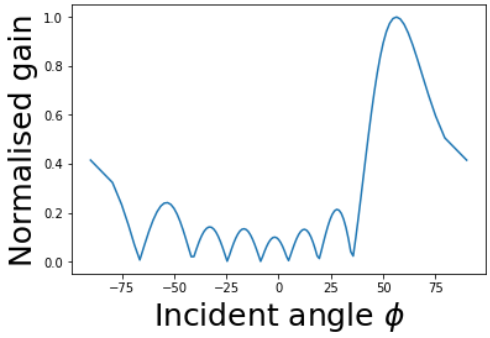
Although the transmitted signal is pulses of real sinusoid $\sin(\omega t)$, where $\omega = 2\pi \cdot 40000$, the matched filter for phase shift and MVDR beamformed signals must have complex coefficients $e^{j\omega t}$. This ensures the matched filter to pass a single frequency component at ω, and block the negative frequency counterpart. It is necessary because the phase delays needed in phase shift beamforming are dependent on the frequency as shown in Equation 3.14. Therefore, phase shift and MVDR beamforming algorithms are correctly applicable only to a single frequency component, which we chose to be the positive frequency ω.

This effect is better illustrated in Figure 5.6 using three images that show the result different matched filters applied to a phase shift beamformed signal. The tested signal is a 40 kHz sinusoid generated using the code in Appendix B. Eight channels were incrementally delayed with 50 samples, to simulate a signal

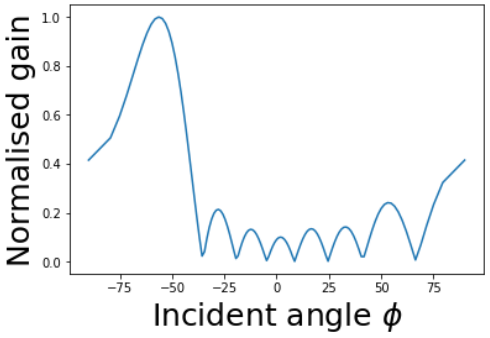
received by a row of microphones with incident angle of 56.4°. Therefore, a correctly beamformed result should show a single main lobe at an incident angle of 56.4°. Figure 5.6a shows the beamformed signal filtered with real sinusoid matched filter. The figure shows two peaks, at ±56.4°. This is caused by the matched filter letting through negatively beamformed signal that is shown in Figure 5.6c. By correlating with a complex signal $e^{j\omega t}$, the image can be filtered and the result shown in Figure 5.6b shows the correctly beamformed gain.



(a) Correlation with $\sin(\omega t)$.



(b) Correlation with $e^{j\omega t}$.



(c) Correlation with $-e^{j\omega t}$.

Figure 5.6: Normalised gain of phase shift beamformed 40 kHz sinusoid signal with incident angle of 56.4°, correlated with varying complex sinusoids.

Clearly, it is necessary to use a matched filter with complex weights to identify the positive frequency signal component and suppress the mirrored image caused by the negative frequency component. Therefore, a matched filter with $e^{j2\pi\frac{40000}{f_s}t}$ as a template will be applied for all beamformed signals and the normalised magnitude will be used to measure the gains of the beamformers. This matched filter serves both as a matched filter for pulse detection and a complex band pass filter to remove the negative frequency component from the beamformed signals.

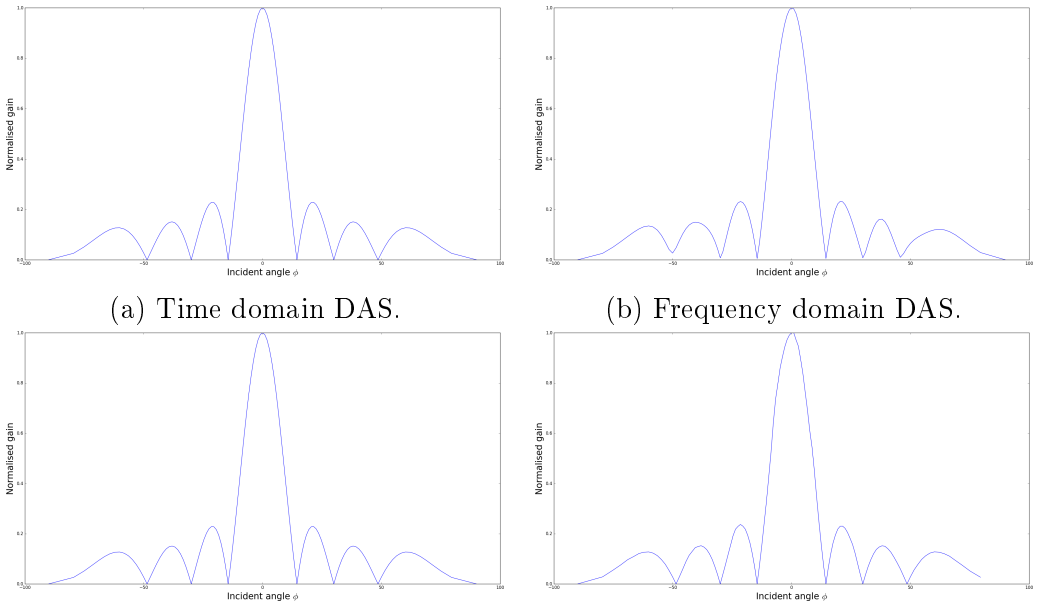
5.4 Comparison of digital beamforming methods

Beamforming in essence filters a signal by applying delays and gains to the signals capture by the individual elements of an array. While there are various ways to implement these delays and gains while yielding mathematically identical outcomes, each implementation comes with varying computational efficiency and limitations. In this section, alternative methods will be implemented in Python and compared to choose the most suitable beamforming method.

Delay and sum (DAS) beamforming is the most straightforward beamforming algorithm and requires only delays to be applied to the signals captured at each array element before summing. The mathematical formulation of DAS beamforming was discussed rigorously in Section 3.5, and provided predictions on its gain patterns and the performance in terms of angular resolution. This is confirmed by implementing DAS beamforming in Python and testing on a synthetic signal. The test signal is a 40 kHz sinusoid with no phase difference between the receivers, thereby signifying a far-field signal with an incident angle of 0° in a noiseless environment.

A signal delay can be achieved in both the time and the frequency domain, by using delay lines in the time domain or through multiplying a linearly varying phase in the frequency domain, while producing identical results. Figure 5.7a and Figure 5.7b shows the gain pattern of the beamformer using the two delaying methods. The time domain DAS was implemented by delaying the PDM signals before demodulating. This allows the algorithm to take advantage of the PDM's high sampling rate, and shows the maximum angular resolution achievable without incorporating interpolation. On the other hand, frequency domain DAS beamforming was applied after the demodulation to reduce the number of samples, thereby reducing the computational load of the FFT. Both of the gain patterns in Figure 5.7a and Figure 5.7b correctly show the direction of the source with its main-lobe located at the centre where it indicates the incident angle of 0°. While the resulting gain patterns are identical, frequency domain DAS beamforming is functionally superior, as it can apply arbitrary delays to the signals. However, frequency domain DAS beamforming is computationally more expensive due to the FFTs involved in the computation. The run times of the two methods are compared in Table 5.1. While execution times can vary widely depending on the hardware and implementation, the table shows that the frequency domain DAS takes over 15 times longer than time domain DAS to compute.

In order to achieve fast computation without necessitating interpolation, the delays can be approximated as phase shifts by assuming narrowband signals. Phase shifts can easily be implemented as a multiplication in the time or the frequency domain with a constant complex value. The gain of a phase



(a) Time domain DAS.

(b) Frequency domain DAS.

(c) Phase shift beamforming.

(d) FFT beamforming.

Figure 5.7: The gain pattern of the beamforming algorithms applied to a signal arriving with an incident angle of 0°. The main-lobe at 0° shows the signal's angle of arrival. The gain pattern from a single source coincides with sinc function as predicted from Equation 3.16.

Beamforming method	Time taken (s)
Time domain	0.13
Frequency domain	2.14
Phase shift	0.031
FFT based	0.015
MVDR	0.047

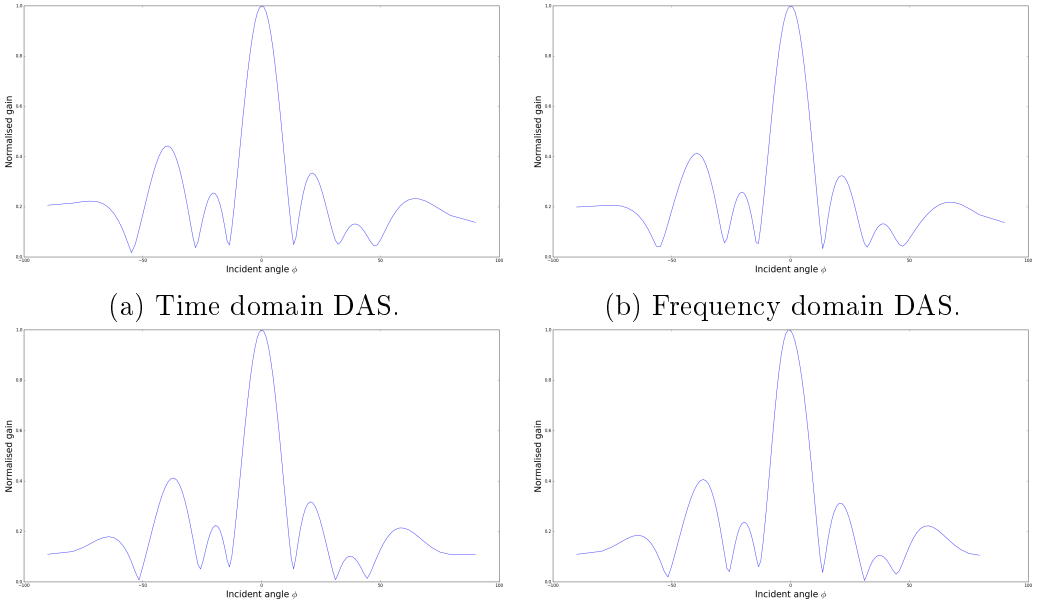
Table 5.1: Average time taken to perform 8 element linear array beamforming.

shift beamforming array is shown in Figure 5.7c and it is again identical to the other methods. Furthermore, phase shift beamforming can be optimised as a spatial FFT, since the computation of phase shift beamforming is identical to the steps of a DFT, as shown in Section 3.5. The result of this FFT based implementation is shown in Figure 5.7d, and is again identical to the other methods. The execution times for phase shift and FFT based beamforming are also shown in Table 5.1. The result shows that the phase shift beamforming significantly reduces the computation time, and there is a further reduction when using the FFT.

The plots in Figure 5.7 are identical to the gain pattern produced based on the mathematical model for 8 element array, shown in Figure 3.6g. This proves the accuracy of the model and therefore other predictions made based

on the model. For example, the theoretical angular resolution achievable using the 8 element linear array is 6.36°, as predicted from Equation 3.18.

Note that because a delay is identical to a linearly varying phase shift where the amount of the phase shift is dependent on the frequency, applying a constant phase shift can only be identical to a delay for a narrowband signal, such as the sinusoid used in the simulation. In real situations, the captured signals inevitably include some broadband noise. Adding uncorrelated noise to the signal and performing the same beamforming functions results in Figure 5.8. The signal arriving with zero incident angle is still present and clearly visible at the centre of the plots. However, the side-lobes are no longer symmetric due to the random noise added to the signal. Notice the results of the two DAS methods shown in Figure 5.8a and Figure 5.8b are identical to each other, while the other two phase shift based methods shown in Figure 5.8c and Figure 5.8d are different. The difference is small, but it can be observed at the left edge of the graphs. The deviation is minor and it only appears when there is a significant amount of noise. However, this shows the limitation of approximating delays as phase shifts when dealing with a broadband signal.



(a) Time domain DAS.

(b) Frequency domain DAS.

(c) Phase shift beamforming.

(d) FFT beamforming.

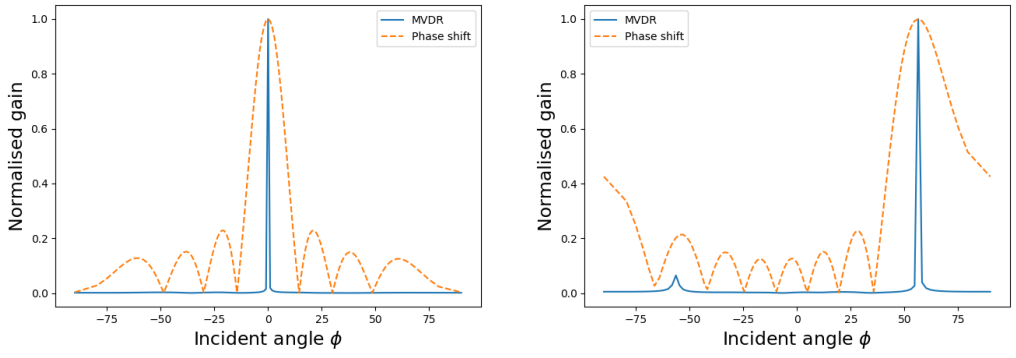
Figure 5.8: The gain pattern of the beamforming algorithms on a signal with noise. Y-axis: normalised gain, x-axis: incident angle.

Implementing various beamforming methods and testing them on synthetic signals allowed us to validate the beamforming software and compare the efficiency of the algorithms objectively. The outcomes of beamforming software coincided with the mathematical models, thereby confirming the model and the software implementation. The tests showed that the FFT based beamforming

is an order of magnitude faster than other beamforming methods. However, FFT beamforming is rigid as the individual weights cannot be controlled when utilising pre-optimised FFT libraries. Meanwhile, phase shift beamforming allows easy configuration of the delays and the gains, as the amount of delay and gain can easily be manipulated in terms of a complex valued weight vector. This flexibility of phase shift beamforming is necessary when optimising the beamformer, especially when implementing adaptive beamforming algorithms. As phase shift beamforming provides a good balance between the computational efficiency and the beamforming flexibility, it is chosen as the beamforming method of choice from the four possible methods described above.

5.5 MVDR beamforming

Minimum Variance Distortionless Receiver (MVDR) is an adaptive beamforming algorithm that can minimise the effects of uncorrelated noise during beamforming. The implementation of a MVDR beamformer is similar to that of the phase shift beamformer, with an extra step to calculate the optimal weights vector as shown in Section 3.6. As was done for DAS and phase shift beamforming, MVDR beamforming is tested on a synthetic signal before being used on a real measurement. This allows the efficacy of the algorithm to be objectively tested. In order to compute the MVDR weights for a set of signals obtained from an array receiver, the signals need to be independent of each other. In practical situations, this is usually the case as the signal includes random noise originating from the microphone's pre-amp and the environment. For the simulation, the code in Appendix B generating 64 sinusoid channels is altered to include random and uncorrelated noise for each channel.



(a) Source at $\phi = 0°$. (b) Source at $\phi = 56.4°$.

Figure 5.9: The gain pattern of the MVDR beamformer (blue solid line), compared with the gain pattern of a phase shift beamformer (orange dashed line).

Figure 5.9a compares the gain pattern of the MVDR beamformer (solid blue line), with the gain pattern of the phase shift beamformer (orange dashed line) when a signal is received with an incident angle of 0°. The graph shows that the gain pattern of the MVDR beamforming has extremely narrow beam-width and completely attenuated side-lobes. The angular resolution achieved by the MVDR beamformer exceeds the theoretical limit attainable using a conventional phased array beamforming and it is known as super-resolution. The gain pattern is plotted with 1 sample delay resolution as it was done for the other beamforming methods above and took 0.047 seconds on average to compute. The time taken is longer than the 0.031 seconds phase shift beamforming has taken to compute for the same resolution, as expected from the extra computation required to calculate the optimal weights vector. However, compared to the other beamforming methods listed on Table 5.1, it is still computationally more efficient.

The merit of MVDR beamforming is even more pronounced when the incident angle of the source is off the centre as shown on Figure 5.9b. The gain pattern of the DAS or phase shift beamforming has a period of π radians, with the most suppression occurring at $\frac{\pi}{2}$ radians away from the incident angle. This causes an emphasised side-lobe to appear for a signal with large incident angle. However, MVDR beamformer is designed to minimise any signal that does not originate from a given incident angle, and maintains almost completely flat gain pattern with no clearly visible side-lobes. Based on these results, MVDR beamformer appears to be clearly superior to the other considered beamforming methods.

While MVDR beamformer is very effective in rejecting uncorrelated noise, it has limitations when dealing with other forms of noises or interferences. To illustrate this, Figure 5.10 shows the gain pattern of phase shift and MVDR beamformers with varying amount of Gaussian noise added to sinusoidal signals. The graphs in the left-hand column shows the gain patterns when the received signal originates from a single direction with an incident angle of 30°. As noted previously, in this scenario the MVDR beamformer results in a beamforming performance and noise rejection that is superior to the other considered beamformers. The performance of MVDR degrades as the noise amplitude increases, which is shown by the widening beam-width in the figures. However, the gain pattern converges to that of the phase shift beamformer. This shows that in an environment in which the additive noise is uncorrelated, MVDR beamforming always produces a superior result.

In contrast, the graphs in the right-hand column of Figure 5.10 consider multi-signal scenario with an additional sinusoidal signal received with an incident angle of $-9.5°$. While the gain pattern of the phase shift beamformer exhibit a sum of two gain patterns whose peaks correctly identify the two incident angles, the gain pattern of MVDR beamformer shows a false peak on the left. Interestingly, the false peak of the MVDR beamformer decreases with more additive uncorrelated noise. Figure 5.10b, Figure 5.10d, and Figure 5.10f

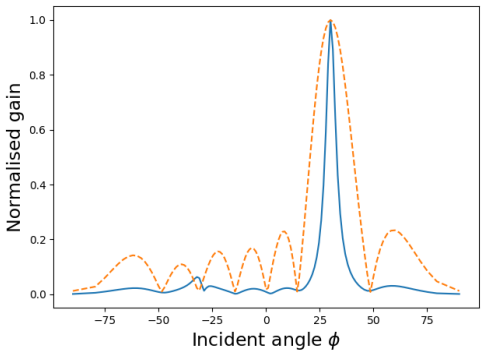
shows how the falsely created image gets suppressed with the increasing noise power.

This is a known limitation of MVDR and other adaptive beamformers, where the performance of the algorithms are severely degraded in the presence of highly correlated noise (Reddy *et al.*, 1987). This is caused by the basic assumption of adaptive beamformer designs, that the interfering signals are assumed to be uncorrelated with the desired signal. Therefore the degradation is worst for coherent signals (Correlated signals that are amplitude-scaled and phase-shifted version of one another) that are used in simulation for Figure 5.10. Such correlated interference can cause the beamformer to fail to create nulls where it is supposed to, or to distort the signal that it was designed to pass. By adding uncorrelated noise, the correlation between the signals is reduced, explaining why adding more noise suppressed the false image. The idea of reducing correlation has been used to improve the algorithm through spatial dithering, which physically moves the array perpendicular to the direction it is facing, thereby reducing the correlation between the interferences while the desired signal does not get effected (Cho and Ahmed, 1987). Another solution, called spatial smoothing, can also be used and has been shown to separate multiple sources while using MVDR beamforming (Zoltowski, 1988). However, even with the application of spatial smoothing, the number of coherent signals that an array can resolve is limited to $\frac{2}{3}M$, where M is the number of sensor elements (Du and Kirlin, 1991). This unfortunately makes the MVDR beamforming unsuitable for imaging, since an image can in general be considered to be made up of a very large and unknown number of targets. We cannot predict the number of targets reflecting the signal in a scene, and therefore a robust beamformer that can function regardless of the possible coherent noises is needed. Such design based on MVDR is complicated and exceeds the scope of this research. Since phase shift beamforming is robust to the number of targets and can achieve the anticipated angular resolution, this type of beamforming will be chosen for our system.

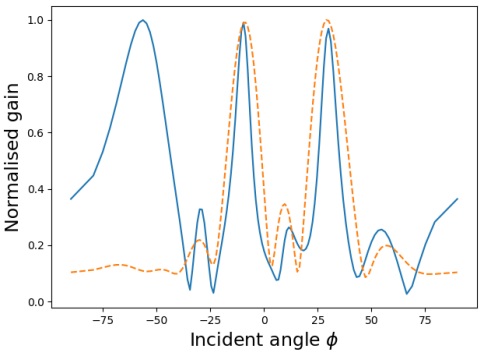
5.6 3D beamforming simulation

After testing various beamforming methods, phase shift beamforming was chosen as the most suitable implementation for the beamforming array. To simplify debugging and visualisation during development, all beamforming methods so far were tested on a linear array only. Linear array beamforming allows the localisation of an object in 2D space in terms of of incident angle and distance. By fully utilising the array hardware and expanding the beamforming to the 2D planar array, an object can be localised in 3D space.

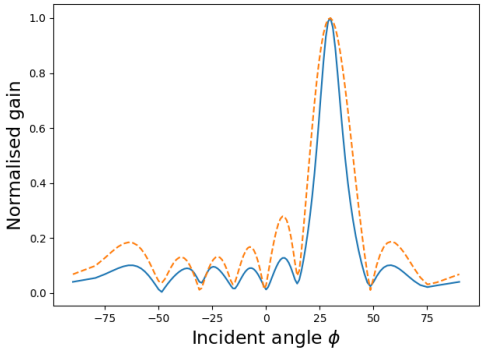
The expansion of linear array design to a planar array was discussed in Section 3.7. The resulting mathematical model for a planar array's gain pattern is shown in Figure 5.2. As we have already tested the conformance of a



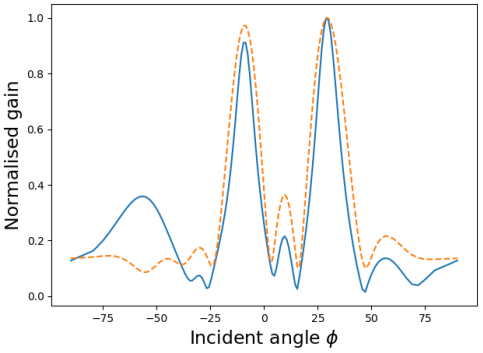
(a) Source at $\phi = 30°$, uncorrelated noise with $\sigma = 0.1$.



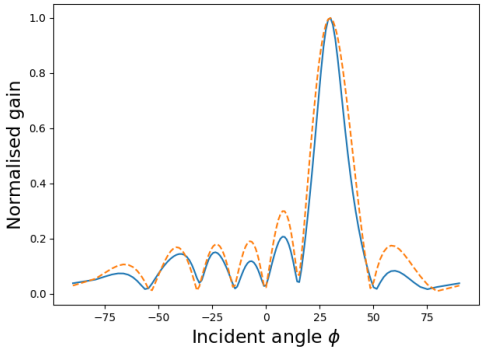
(b) Sources at $\phi = 30°$ and $\phi = -9.5°$, uncorrelated noise with $\sigma = 0.1$.



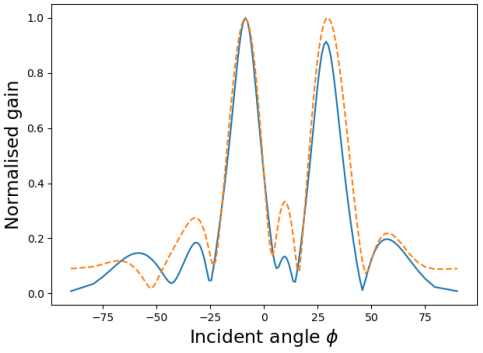
(c) Source at $\phi = 30°$, uncorrelated noise with $\sigma = 0.3$.



(d) Sources at $\phi = 30°$ and $\phi = -9.5°$, uncorrelated noise with $\sigma = 0.3$.



(e) Source at $\phi = 30°$, uncorrelated noise with $\sigma = 0.5$.



(f) Sources at $\phi = 30°$ and $\phi = -9.5°$, uncorrelated noise with $\sigma = 0.5$.

Figure 5.10: The gain patterns of the MVDR beamformer with Gaussian noise added. In each graphs, the blue solid line shows the gain pattern of the MVDR beamformer and the orange dashed line shows the gain pattern of phase shift beamformer that is used as the benchmark.

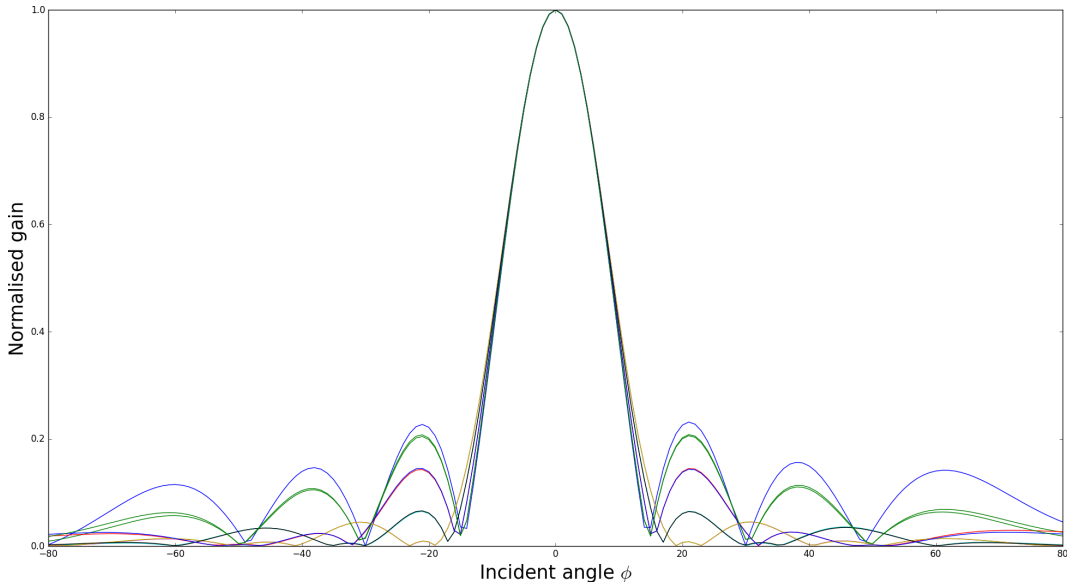


Figure 5.11: Gain pattern of 3D phase shift beamforming with varying azimuth angles from 0° to 90° with 10° intervals.

phase shift beamforming to the theory using a linear array, we only have to ensure that the expansion into a planar array works as predicted as well.

The gain pattern of the planar microphone array beamforming tested on a simulated signal is shown in Figure 5.11. The plot shows the gain patterns along varying azimuth angles, with phase-shift beamforming applied to a far-field signal originating from the direction normal to the array along the Z-axis. Figure 5.11 shows that while the side-lobes may fluctuate depending on the azimuth angle, the beamwidth of their main-lobes are constant. This means the angular resolution of the 8×8 square beamformer is circularly symmetric and identical to that of 8 elements linear array. The fluctuating side-lobes and the circularly symmetric main-lobe beamwidth is consistent to the gain pattern observed in Figure 5.2, that was produced from the mathematical model. Therefore we have confirmed that the phase shift beamforming software works as predicted from the mathematical model.

5.7 Conclusion

The microphone array beamforming software has been developed and tested through the use of simulated signals in this chapter. The outcomes were compared against the mathematical models derived in previous chapters and were validated. Implementations of DAS beamforming and phase shift beamforming were compared, and phase shift beamforming was chosen for its flexibility and efficiency. Testing MVDR beamforming revealed its limitations in the presence of coherent noises, which made it unsuitable to be used for beamforming for

gesture recognition.

Beamforming software can now be applied to a real signal captured by an array and produce a 3D gain pattern at any point in the signal. By analysing the direction of the main-lobe in gain patterns, the distance and orientation of the signal source can be found. Next chapter will apply the software to real data captured using the microphone array and visually analyse the data.

Chapter 6

Data capture

Once both the hardware and software required for sonar imaging was developed and tested using artificial signals, the microphone array hardware was used to take real measurements. Hardware and software were first tested by taking some simple, static measurements and comparing these against simulated predictions. Once the functionality of the system was affirmed in this way, the set-up was used to capture a set of 8 gestures. This captured data was processed further for classification.

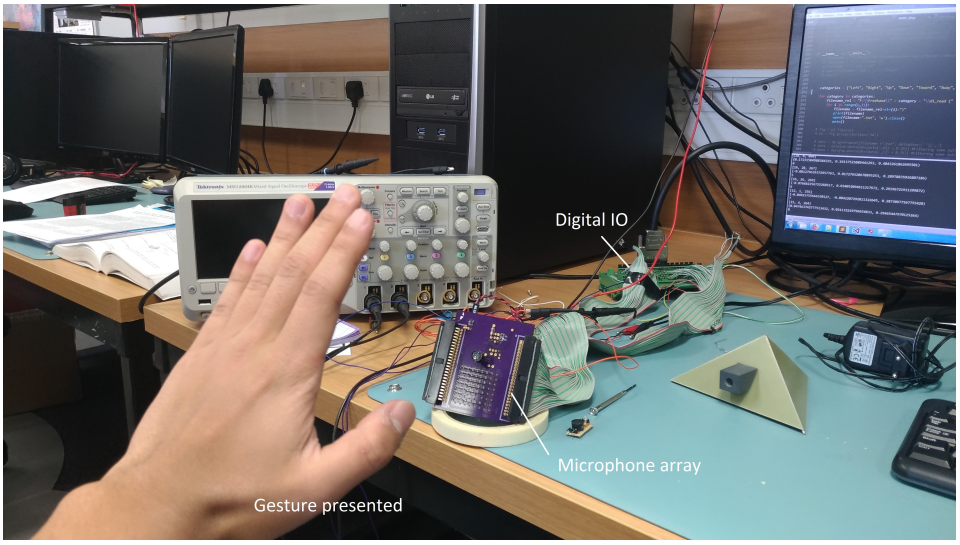


Figure 6.1: Hardware set up for the measurements.

There are two orientations in which the microphone array can be easily placed on a desktop. The first and more obvious is flat on the table, so that the Z-axis is directed vertically upwards. This orientation avoids clutter from the desk or other surrounding environment. However, it was discovered that this orientation suffers from an awkward placement of the capture area. The

volume microphone array can reliably capture gestures is limited by the dead-zone and the unambiguous range. If a gesture is performed too close to the array (in the dead-zone), the microphones are saturated by the transmitted signal and the reflected signal cannot be discerned. On the other hand, if the gesture is performed too far (further than the unambiguous range), consecutive pulses are transmitted before the return echo is received, making the distance measurement unreliable. The microphone array hardware has a dead-zone of 0.2 m and an unambiguous range of 1.2 m. While the unambiguous range did not lead to difficulties, having to maintain a minimum distance of 0.2 m above the desk while presenting gestures was awkward and difficult. Therefore, during data capture, the microphone array was set up vertically on the edge of the desktop as shown in Figure 6.1. Using this set up, the user can choose the distance from the microphone by moving forward or backward, instead of by lifting the hand. This orientation also allowed the user to utilise more space when presenting gestures while seated. By placing the microphone array on the edge of the desk, the clutter from the desk was also avoided.

The positioning of the microphone array also affects the orientation of the captured images after beamforming. While the positive Z-axis is always the direction away from the array, the XY-axes depends on the organisation of the microphones within the beamforming software. When facing the microphone array, the user's right hand side is chosen to be the positive X direction, and the top side of the array becomes the positive Y direction.

6.1 Beamforming gain

Before proceeding with data acquisition, the hardware was tested with a real signal and the results were compared against the predictions made using simulations. To ensure a clear and reproducible signal reflection, a trihedral corner reflector was constructed with 10 cm long edges. The orthographic projection of the reflector's design is shown in Figure 6.2. This reflector is used unless the target is otherwise specified, thereby ensuring consistency of the captured signals.

In order to test the beamformer gain, the target was placed near the centre of the array and the gain of returned signal is plotted as shown in Figure 6.4. The solid blue line shows the gain pattern from the measurement, while the dotted green line shows the gain pattern using a simulated signal. The two plots show coinciding beam-width, nulls, and side-lobes. This confirms that the simulation accurately predicts the real measurement.

Beamforming was tested on many other signals with varying target locations in order to ensure the accuracy of the beamformer implementation. The results were in all cases consistent with the simulated results, confirming the models and the implementation of the beamforming array. However, plots of gain patterns are insufficient to provide additional insights about the captured

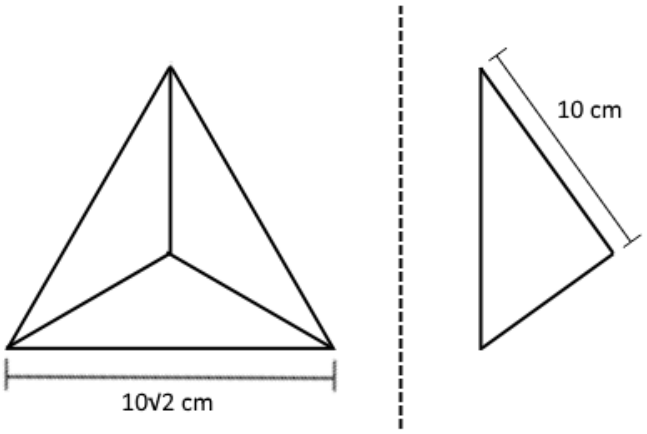


Figure 6.2: The orthographic projection of the corner reflector.

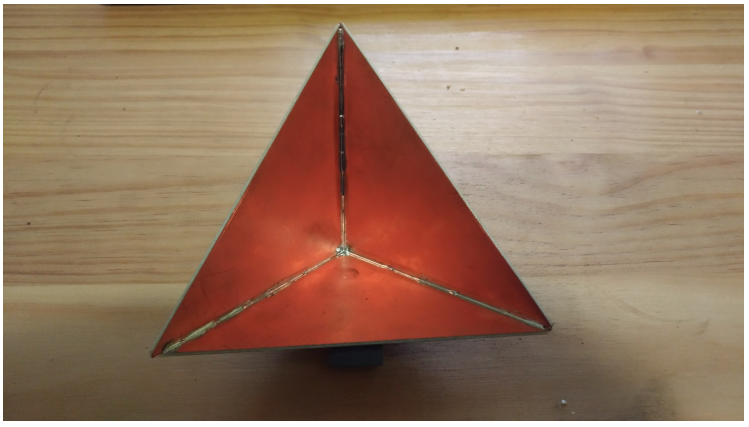


Figure 6.3: The trihedral corner reflector used as a target.

data, as it only shows the gain of the beamforming array. In order to better visualise the captured signals an B-mode image representation of data is used.

6.2 B-mode imaging

B-mode is short for brightness mode and normally used for referring to a 2D image created from a scan by a linear array. The two axes of B-mode image represents the displacement along the array and the broadside distance, with the intensity of the signal that originates from the given point represented by the colour of the image. To illustrate the 3D data from the planar array in a 2D image, the images will be sliced along a specific azimuth angle that best illustrates the captured data.

B-mode imaging was first tested with the target slowly approaching the centre of the array. Figure 6.5 shows the series of images captured at 15 fps.

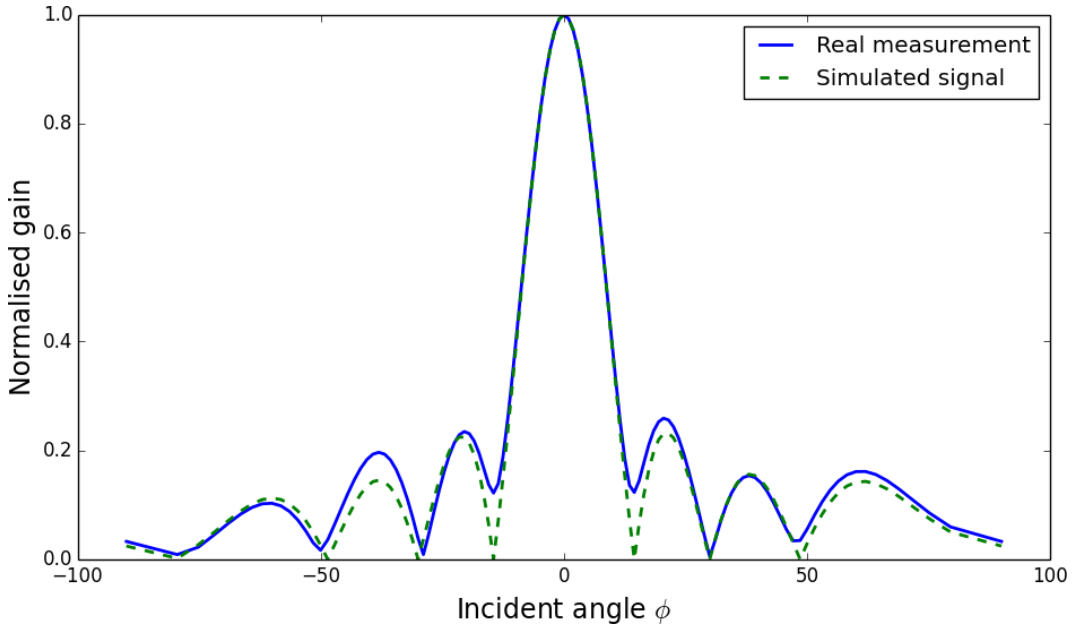


Figure 6.4: Measured signal vs simulated signal's gain patterns.

The images show slice of 3D data through the x-axis, with the azimuth angle zero. The y-axis of the figures shows the incident angle between $-80°$ to $80°$, and the x-axis shows the distance, with the maximum distance corresponding to the unambiguous range of 1.2 m. The top image shows the initial position of the target to the far right. The following images indicate the target progressively moving toward the array. The series of B-mode images clearly demonstrates the capability of the beamforming array for localising a single target's direction and the distance.

Each of the images in Figure 6.5 shows a strong signal close to the microphone array. This is the signal within the dead-zone and is due to direct signal propagtion from transmitter to microphones. The high amplitude of the signal captured in the dead-zone overwhelms the signal of interest and makes it hard to separate the target. This can be avoided by cropping the first 0.2 m of the data and thereby remove the dead-zone as shown in Figure 6.6.

Note also that the noise in the dead-zone appears worse than it is because the images did not take polar coordinates into consideration. A grid based on polar coordinates gets finer towards the array, and as a result objects closer to the array appear larger than it is. Figure 6.7 shows correctly scaled images of the first and last frames of Figure 6.5. Even without cropping, the dead-zone noise appears as a tiny, focused dot that can easily be ignored. While using polar coordinates produces more accurate scaling for the noise in the dead-zone, they also cause the same target to appear in different sizes depending on the distance. This is because while a target is captured with a constant angular resolution, but polar grid represents them differently depending on the distance. This is evident from Figure 6.5 and Figure 6.7, where they

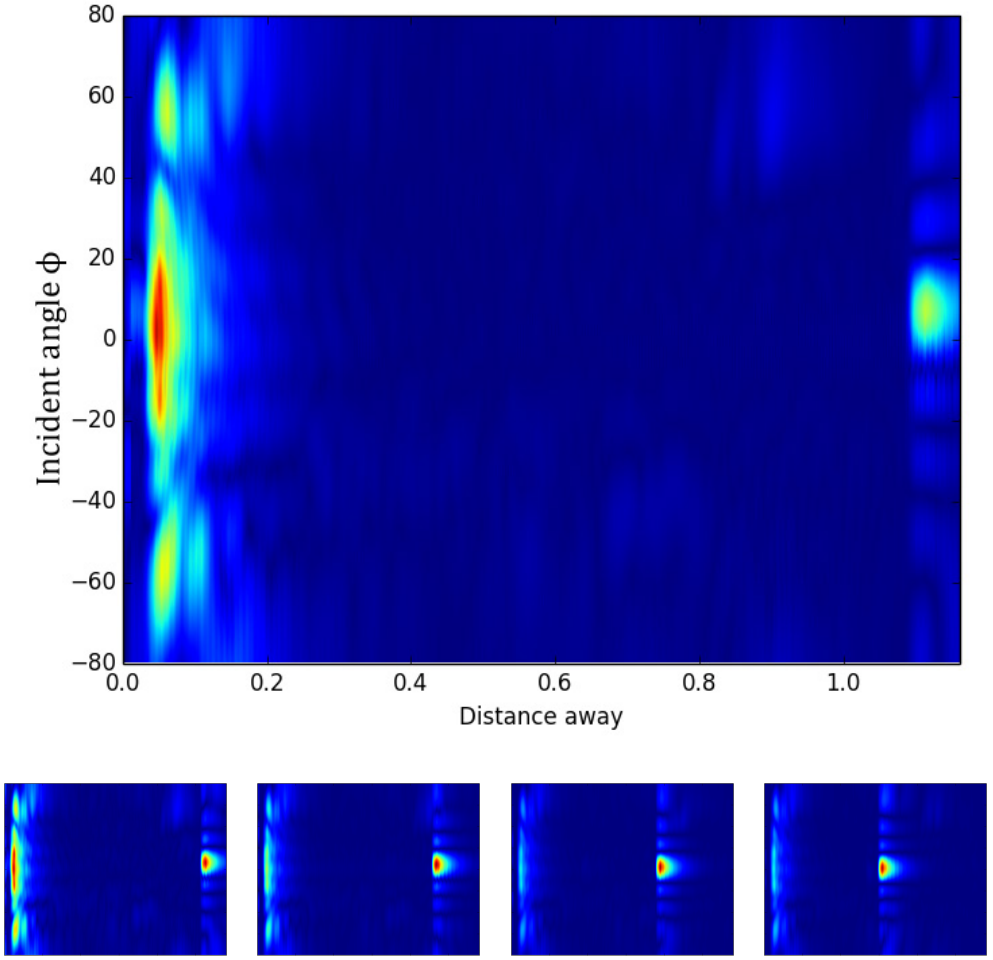


Figure 6.5: A target (bright spot on the right) moving towards the array on the left. The images progression begins at the top, then from left to right.

show the two representations of the same data. While Figure 6.5 represents the target in similar sized marks throughout the images, the two images in Figure 6.7 show the target with vastly different sized marks. The effect of changing size is worse for visualisation as it gives a false impression of changing image resolution. Since the dead-zone noise can easily be removed from the images, the rectangular representation of the B-mode image as used previously is preferred to the polar representation.

The next image shows a capture with two hands raised on the either side of the torso, placed slightly forward. The result in Figure 6.8 shows the clear separation between the torso (highlighted in a purple box) and the two hands (two red boxes). This result shows that the beamforming array is capable of capturing body parts without using the reflector as a target and it is able to separate them. The image was produced twice, once using a single row of the array and another utilising the full 8×8 planar array. The result shows that

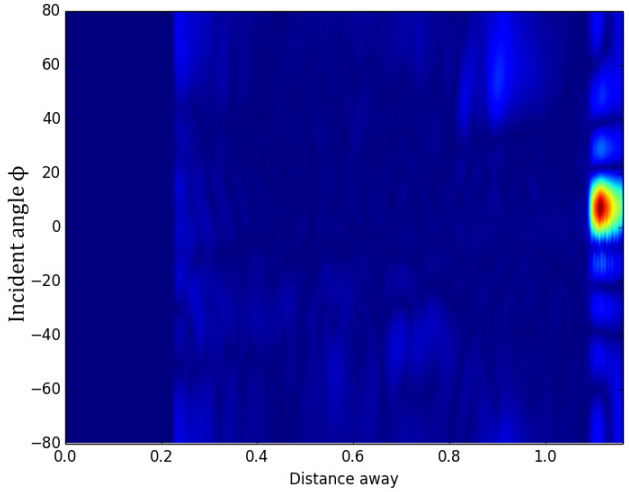


Figure 6.6: The first image in Figure 6.5 with the dead-zone cropped out.

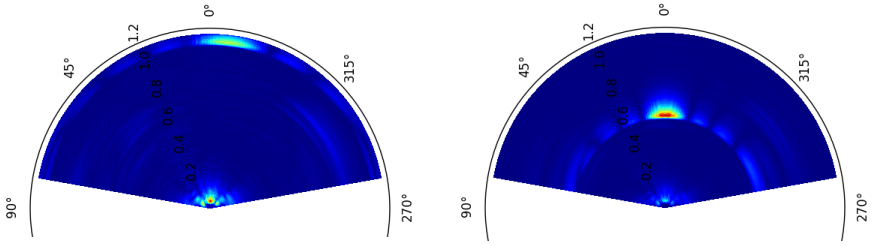
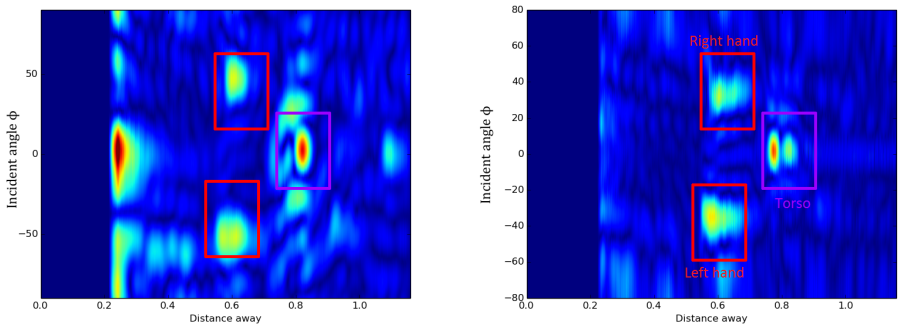


Figure 6.7: The first and last images in Figure 6.5 using polar plots.

the advantage of using an 8×8 planar array is minor in terms of the resolution. While a small improvement can be observed in the quality of planar array beamforming shown in Figure 6.8b over Figure 6.8a, the improvement is observed in the form of reduced noise and the sizes of the beamformed objects are not notably smaller when using the planar array. This observation is consistent with the result shown in Equation 3.18, which tells us the beamforming resolution is only dependent on the number of components along the beamforming axis. Adding a new dimension to the array only adds the capability to beamform in the new dimension and does not necessarily produce a better image. However, we can still expect a better noise rejection from using the average of multiple linear arrays when using a planar array, as confirmed by the results in Figure 6.8b.

Finally, Figure 6.9 shows another B-mode image, cross-sectioned in Y-Z axis. This orientation shows the distance and the relative height of the object, instead of its location in X-axis. From the size of the reflected signal represented on the image, it can be observed that the angular resolution in Y-Z axis is similar to that in X-Z axis. Indeed, the beamforming performance is identical in X and Y axis due to the symmetry in design. Therefore, we can treat the Y axis as being identical to the X axis. The additional axis allows us



(a) 8 element linear array. (b) 8×8 planar array.

Figure 6.8: Two hands raised on head level.

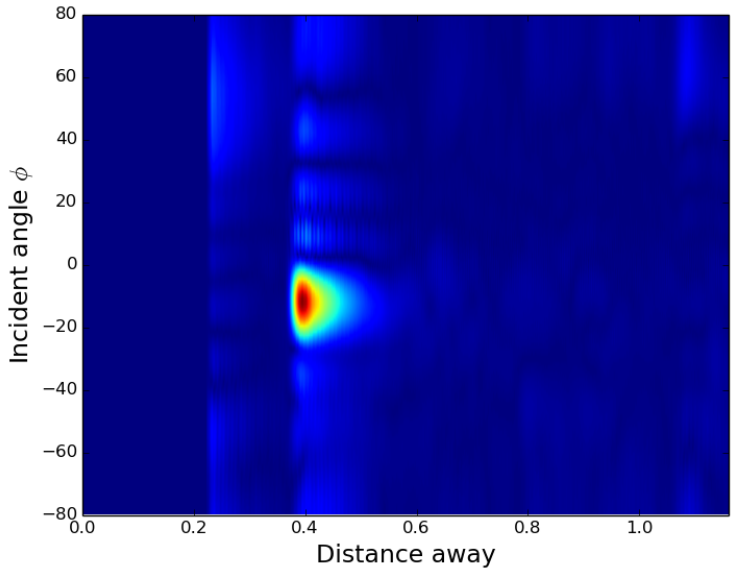


Figure 6.9: Cross section through Y-Z plane, with the azimuth angle of 90°.

to localise the object reflecting an echo in 3D space. The 3D location of the objects captured is used to identify gestures in the next chapter.

6.3 Gesture capture

The main advantage of using a planar array for gesture recognition is being able to capture objects' locations in 3D. While linear ultrasound array with similar capabilities were tested for gesture recognition in other studies, the ability to capture 3D gestures provides an unique advantage and flexibility to our system. In order to test its capability, a set of simple gestures that can utilise the 3D imaging capability were chosen as test cases.

A swipe gesture, where a hand moves in a single direction, is a often used

gesture that is simple to recognise. Swipes in different directions may hold different meanings. Therefore a gesture recognition system must be able to discern the direction of the swipes in order to correctly recognise a gesture. Hence a set of swipe gestures - up, down, left, right, forward, and backward - is used to test the gesture recogniser's ability to recognise the movements of hand in different directions. To test more advanced cases, two circular gestures in clockwise and anti-clockwise rotation are added. These include movement in all directions, while being notably different from the swipes and contrasting each other. Therefore this set of 8 gestures provide realistic, easy to differentiate set of test cases that can be used for testing the system. Each of the eight gestures was repeated 20 times and the collected data is used as the basis for developing and testing the gesture classification.

The test cases are purposefully limited to single handed gestures, thus simplifying the feature extraction process. While it was shown in Figure 6.5 that the beamforming array is capable of producing images where the different body parts can be visually separated, recognising multiple body parts and classifying to a correct gesture presents a complicated problem. While it is possible to use a pre-existing image recognition libraries to achieve some of this, it was decided to simplify the dataset instead. The process of extracting the useful features is discussed further below.

6.4 Conclusion

The microphone array was used to capture real images and the data was process to create various visualisations that validates the design of the microphone array. The results were compared with the corresponding simulations to confirm the accuracy of the simulations and also the implementation of the beamforming array. Once the design of the microphone array was validated, the data was used to create B-mode images. The B-mode image visually showed that the capabilities of microphone array, that it can correctly capture objects in 3D to the point a human can discern the object's location and its motion.

With the design validated, a set of gestures were presented and recorded for the final stage of gesture recognition - classification. The next chapter will discuss the techniques used for gesture recognition in three steps.

Chapter 7

Gesture recognition

The previous chapters have described the development of hardware and software that allows hand gestures to be captured. 3D images of gestures were captured using the 8×8 beamforming microphone array whose design and construction was described in the previous chapters. To test the viability of automatic classification, 8 simple gestures are chosen as the test cases. The chosen gestures are: moving a hand 1) up, 2) down, 3) left, 4) right, 5) towards, and 6) away from the array; and drawing a circle in 7) clockwise and 8) anticlockwise directions. The set of gestures were captured 20 times each using the corner reflector to enhance the SNR of captured images.

In order to recognise the gesture, data capture must be followed by feature extraction and then classification, as described by (Berman and Stern, 2012). Feature extraction, as the name suggests, extracts the distinguishing information of the presented gesture from the captured images. In our case, feature extraction will represent each gesture by a time series of 3D points that can be visualised as traces. The traces are then compared with templates for classification. This chapter discusses the process of extracting the traces and of comparing them with the templates.

The classification of hand trajectories has been approached in four ways by other researchers: support vector machine (SVM), nearest neighbour (NN), hidden Markov model (HMM), and dynamic time warping (DTW) (Cheng *et al.*, 2016*b*). Time did not permit different techniques to be comparatively evaluated in this work. Instead, an approach using a nearest neighbour classifier in conjunction with DTW is used to match gestures based on the templates. Specifically, DTW with 1NN (1 nearest neighbour) is known to be an effective algorithm to use when classifying time series data (Xi *et al.*, 2006; Mitsa, 2010). While the combination is computationally expensive, it requires little training data while allowing high accuracy and has been extensively used for gesture recognition (Cheng *et al.*, 2016*a*; Plouffe and Cretu, 2016). The simplicity of the algorithm and high performance makes it well suited to be used on proof of concept development on a small dataset.

7.1 Feature extraction

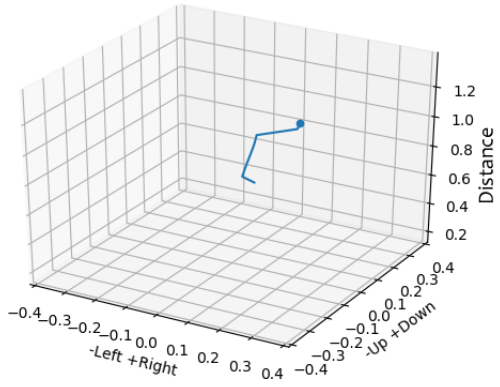
While the raw 3D images can be used to visualise the gestures, they contain a large amount of redundancy that obscures useful information. Feature extraction reduces the data by extracting the descriptive information required for gesture recognition. Most image processing libraries include a number of feature extraction algorithms that usually involves edge or motion detection, and that can be used to identify and track an object within an image. Our research is focused on establishing the capability of the developed 2D sonar array hardware for gesture recognition. Therefore, a simplified feature extraction technique that tracks the strongest signal within an image was used. The employed technique is effective for single-handed gestures with a clear trajectory. If the image is expected to contain multiple objects, such as two-handed gestures, more advanced feature extraction algorithm must be used to identify the objects and track their movements. These are beyond the scope of this project and left for future work. Therefore, during the data capture, the presented gestures were limited to single handed gestures.

By ensuring that there is only a single target during recording, tracing maxima in each image successfully identified the location of the reflector and traced the movement of the hand holding the reflector. The points that were detected outside of the range $x, y \in (-0.35m; 0.35m)$ and $z > 0.2m$ were discarded. Those detections outside of the given range indicate that the reflector is not detected within the scene and therefore that no gesture is present in that image. Identifying a gesture presentation is normally dealt separately using techniques related to continuous gesture detection, however, our heuristic provides a simple means to identify the images containing valid gestures. This forms a part of simplified feature extraction used in this research, that assumes the traces of gestures are extracted.

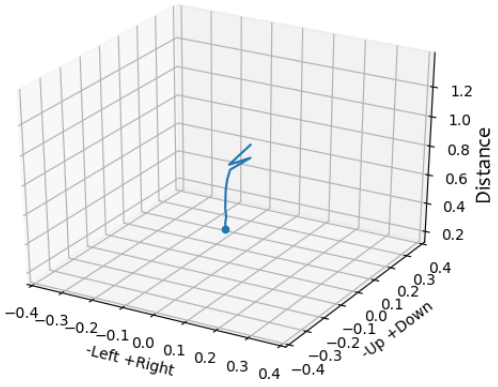
Figure 7.1 shows an example of each of the gestures performed and processed as described above. The dot marks the beginning of the trace. As the gestures were chosen so that they are clearly distinguishable from each other, the gestures can be recognised from observation of the traces. However, computationally matching the traces and classifying them poses another challenge.

7.2 Dynamic Time Warping

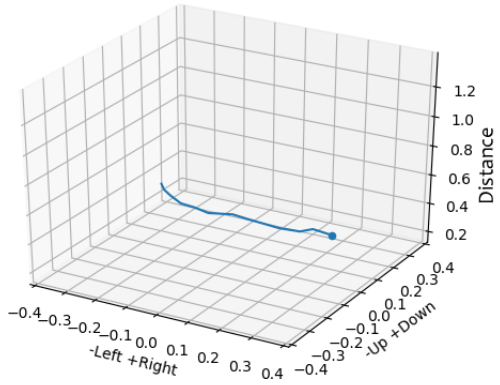
In order to classify gestures, we need a method to measure similarity between the traces extracted from the 3D images. Calculating the Euclidean distance is a simple approach to measure similarities between traces. However, this does not function well when the traces are of unequal length or mismatched in terms of time. We need a more robust algorithm to compare the traces while ignoring other traits such as the time a trace took to complete. Dynamic time



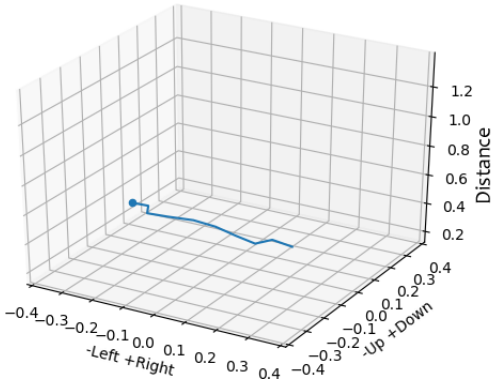
(a) Moving towards the array.



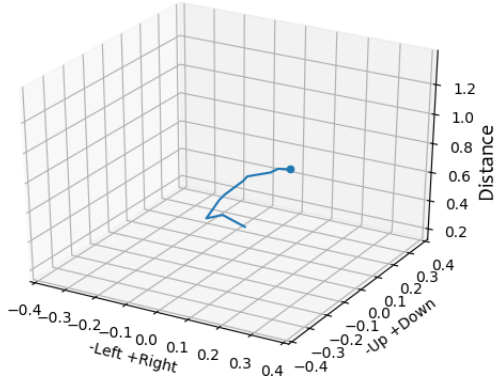
(b) Moving away from the array.



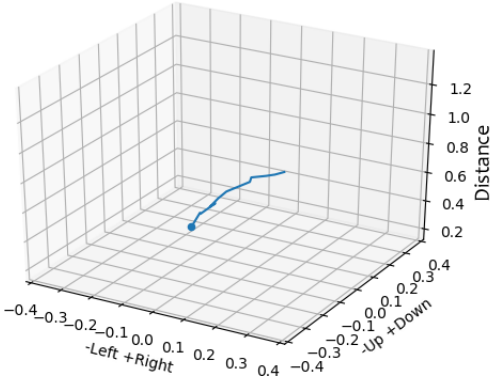
(c) Moving to the left.



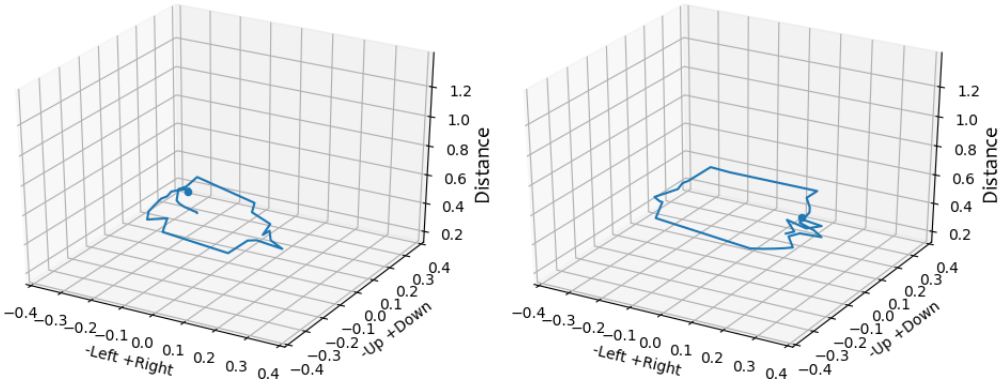
(d) Moving to the right.



(e) Moving up.



(f) Moving down.



(g) Moving in circle, clockwise direction.

(h) Moving in circle, anticlockwise direction.

Figure 7.1: Sample traces for each of the eight gestures. The dot at the end of each traces mark the beginning of the trace. The vertical axis represents the distance away from the microphone array.

warping (DTW) is a well-established algorithm that can be used to measure similarity between two time-series, while ignoring their temporal differences. This algorithm was originally developed for speech recognition, where two records of same utterance may follow the same temporal progression but with varying speed. It has since been successfully used for comparing gestures.

Intuitively, DTW compares two signals while temporally warping them in time (the signals are squeezed or stretched along the time axis) until they are best matched. This allows the algorithm to ignore the distortion in time axis when comparing the two signals. To understand the process, consider two sequences $X = \{x_1, x_2, ...x_n\}$ and $Y = \{y_1, y_2, ..., y_m\}$ of length n and m respectively. We first need to define a function for local distance measure $c(x, y)$, that returns a real valued distance between two points x and y in the sequences, where a smaller value means that the two points are more similar to each other. The DTW algorithm calculates the similarity between two sequences, known as DTW distance, by computing point-by-point local distance measurement of the two sequences while also accumulating the sum.

The process can be understood using a cost matrix of size $n \times m$ that is calculated using local distance measure as shown in Table 7.1. This matrix contains the distance between every point in X and every point in Y. The DTW distance can be calculated by finding a path starting at the left-bottom of the cost matrix (the beginning of both sequences) and ending at the right-top (the end of both sequences), while only moving one step at a time in an upward, right, or diagonal up-right direction.

This is more formally defined as a minimising equation:

$$\min \left(\sum_{i,j \in P} c(x_i, y_j) \right) \tag{7.1}$$

$c(x_1, y_m)$	$c(x_2, y_m)$	$\cdots$	$c(x_n, y_m)$
$\vdots$	$\vdots$	$\ddots$	$\vdots$
$c(x_1, y_2)$	$c(x_2, y_2)$	$\cdots$	$c(x_n, y_2)$
$c(x_1, y_1)$	$c(x_2, y_1)$	$\cdots$	$c(x_n, y_1)$

Table 7.1: The DTW cost matrix.

where P is the warping path the points (x_i, y_j) follow in the cost matrix (Müller, 2007). A warping path is defined by three conditions:

1. Boundary condition: The path begins at x_1, y_1 and ends at x_n, y_m.
2. Monotonicity condition: The values of i and j cannot decrease.
3. Step size condition: The values of i and j can increase at most by 1 at each time step.

The algorithm computing the DTW path first computes the full cost matrix, and then recursively calculates the optimum path. The implementation of DTW for the 3D traces can be viewed in the code attached in Appendix F. In this implementation, the Euclidean distance has been used for the local distance measure.

7.3 K-nearest neighbour

The DTW algorithm can be used to compute a similarity score between the trajectories corresponding to two gestures. By matching a gesture to its most similar template, classification is achieved. The k-nearest neighbour (KNN) algorithm classifies an input based on a set of templates called training set. The input is assigned to the class with the K most similar templates within the training set. If the neighbouring templates differ in class, the input is classified as the most common class among the K neighbours. 1-nearest neighbour (1NN) is a special case of KNN, where a point is classified based on the single closest template in the training set. The 1NN calculates the DTW distance between the input and each of the templates, and chooses the class corresponding to the nearest template.

7.4 Classifier validation

Classification using 1NN algorithm was tested within a 4-fold cross-validation framework. Given 20 repetitions of each gesture (160 trajectories in total), the records are divided into 4 groups with 5 repetitions of each gesture. Each subgroup is then classified using DTW and 1NN method, using the remaining three subgroups as the training set. The result of 4-fold cross-validation of

True gestures \ Predicted gestures	Left	Right	Up	Down	Toward	Away	Clock	Anticlock
Left	1	0	0	0	0	0	0	0
Right	0	1	0	0	0	0	0	0
Up	0	0	0.95	0	0.05	0	0	0
Down	0	0	0	1	0	0	0	0
Toward	0	0	0.05	0	0.95	0	0	0
Away	0	0	0	0	0	1	0	0
Clock	0.05	0	0	0	0	0	0.95	0
Anticlock	0	0	0	0.05	0	0	0	0.95

Table 7.2: Normalised confusion matrix of classification of the gestures presented using the corner reflector, produced by 4-fold cross-validation.

True gestures \ Predicted gestures	Left	Right	Up	Down	Toward	Away	Clock	Anticlock
Left	0.8	0	0.15	0	0.05	0	0	0
Right	0	0.95	0	0	0	0.05	0	0
Up	0	0	0.95	0	0.05	0	0	0
Down	0	0	0	0.95	0	0.05	0	0
Toward	0.05	0.05	0	0.05	0.8	0.05	0	0
Away	0.1	0.1	0	0	0	0.8	0	0
Clock	0	0.1	0	0	0	0	0.9	0
Anticlock	0	0.05	0	0	0	0.05	0	0.9

Table 7.3: Normalised confusion matrix of classification of the gestures presented with bare hand, produced by 4-fold cross-validation.

the classifier on the gestures presented with a corner reflector is normalised and shown in Table 7.2. The columns indicate different predictions, while the rows indicate the true gesture. The correct matches along the diagonal line are highlighted. For example, the third row in Table 7.2 shows that a gesture moving 'Up' has 95% chance to be classified correctly, while 5% chance to be classified falsely as a gesture toward the array. The result shows near perfect classification, with above 90% chance of correctly classifying the gestures. The DTW with 1NN classifier performed well beyond expectation for a proof of concept implementation and classified gestures with high accuracy.

After validating the classifier on the data that was captured using a corner reflector, the same set of gestures were presented again, this time without using the reflector. The captured gestures were processed in the same way as the reflector aided gestures, evaluated using 4-fold cross-validation. The results in Table 7.3 shows lowered accuracy for all gestures. However, the probability of correct classification remains above 80%. This is considered a promising result in view of the fact that tracing the maxima is generally no longer a reliable feature extraction approach when the reflector is not present. We anticipate

True gestures \ Predicted gestures	Left	Right	Up	Down	Toward	Away	Clock	Anticlock
Left	0.6	0	0	0	0	0	0.15	0.25
Right	0	0.15	0	0.1	0	0	0.75	0
Up	0	0	0.35	0	0.05	0	0.25	0.4
Down	0	0	0	0.25	0	0	0	0.75
Toward	0.05	0	0	0.05	0	0	0.65	0.25
Away	0.05	0	0	0.05	0	0.35	0.25	0.3
Clock	0	0	0	0	0	0	0.75	0.25
Anticlock	0.05	0	0	0	0	0	0	0.9

Table 7.4: Normalised confusion matrix of classifying free-hand gestures using reflector aided gestures as templates.

that the accuracy with which bare hand gestures can be recognised can be improved by employing a more robust feature extraction approach.

Finally, Table 7.4 shows the result of classifying gestures presented without the reflector (i.e. bare hand gestures), while using the gestures presented with reflector as the template. In this case, the accuracy is much lower than shown in tables Table 7.2 and Table 7.3. This is a limitation of 1NN classification algorithm, where an accurate template of the gesture must exist for accurate classifications. In Table 7.4 the training and testing conditions of the classifier are different (presented with reflector and bare hand respectively) leading to a mismatch between training and testing traces used for DTW. Since the classification is not resilient against a small change in the way gestures are presented, the system needs to be trained for each new gestures in the same setting as its usage.

7.5 Conclusion

The set of gestures captured using the developed 2D microphone array was classified using DTW as a similarity measure and 1NN as a classifier. The combination of two algorithms provide a powerful classification algorithm, with minimal training data requirement. The classification software successfully classified the set of gestures presented with high accuracy. The result shows that even with fairly simple classification algorithms, the microphone array provides sufficient information about the gesture to be used for gesture recognition.

The classification was first tested on the gestures performed using the corner reflector designed in Chapter 6. The enhanced SNR using the reflector allowed the classifier to identify the gestures with high accuracy. Further testing using gestures performed without the reflector showed that, although the accuracy deteriorated, the classifier was still able to correctly identify the test gesture in more than 8 out of 10 cases. However, the classifier resulted in

significantly lower accuracy when gestures using the corner reflector were used as templates to identify gestures presented without the reflector. This shows a limitation of the current classifier, which is not robust to changes in the way gestures are presented. Therefore, the classifier needs a training set that was recorded in the same condition as the gesture that is to be classified. Alternatively, this limitation can be mitigated by using a more robust feature extraction method.

Chapter 8

Summary and conclusion

The goal of this thesis was to design and test a system that can be used to perform gesture recognition, by a means other than the optical systems that are currently in popular use. After considering number of remote sensing technologies in Chapter 2, ultrasound was chosen as a promising sensing method. The work described in this thesis is, to the author's knowledge, the first report of a successful use of 2D ultrasound beamforming array for gesture recognition.

In Chapter 3, the design parameters for an ultrasound gesture capture system were studied. In order to capture a gesture that is presented in a 3D space, a focused and easily steerable aperture was needed. A 2D microphone array makes it possible to create a highly directional receiver that can be steered in 3D using software, and hence was chosen for gesture capture. By analysing the design parameters, it was understood that the quality of the images produced by an array beamformer is determined by the angular resolution of the beamforming array, which is in turn dependent on the size of the array aperture and the frequency used. However, the distance between two microphones needs to be less than half of the wavelength of the sensing signal in order to avoid ambiguity. Hence it was clear that an array with high angular resolution required small microphones for dense placement, and that the aperture could be widened by increasing the number of microphones. Therefore the size of individual microphones and the number used imposes a physical limit on the angular resolution achievable using the conventional beamforming methods.

Based on the relationship between aperture size and beamforming performance, and considering available hardware components, the microphone array was designed to be an 8×8 square array in Chapter 4. The microphone array was designed to received the echoes of 40 kHz pulsed sinusoid, transmitted 146.5 times per seconds for 1 ms. In order to beamform the 40 kHz signal, the microphones were required to be 4.25 mm apart. From the mathematical models, the achievable angular resolution with this spacing for 8 element linear array was predicted to be 6.36° and through the extension of the model, the same limit in angular resolution was shown to apply to the 8×8 planar array. This angular resolution is not sufficient to differentiate fine details, such as in-

dividual fingers of a hand, but it can be used to track larger body parts. The hardware required microphones with a small footprint and a large bandwidth. The SPH0641LU4H-1 was identified as a suitable microphone to be used in the array. The necessary control and interface circuit for the microphones, along with a pulse transmitter circuit, was designed. The microphone array was manufactured on a PCB and tested.

Chapter 5 presented the development of beamforming algorithms and the testing of these using a simulated signal. The developed beamforming software was verified by comparing the gain measured using a simulated signal with the gain predicted by the mathematical model. In order to overcome the limited angular resolution achievable by phase shift beamforming, MVDR beamforming was implemented and tested as a possible solution. However, it was discovered through simulation that an MVDR beamformer is not robust to the presence of multiple coherent signals and hence it is unsuitable for ultrasound imaging of gestures. Therefore conventional DAS beamforming was used in the final design.

The developed hardware and software was used to capture real signals which were compared with simulated signals in Chapter 6. This validated the theoretical predictions regarding the gain and the angular resolution of a microphone array beamformer in Chapter 3. It was found that the theory, simulation, and real data conformed to each other, thereby validating the design. The captured images showed that the microphone array, as presumed from its angular resolution, provided sufficient resolution to separate larger body parts such as hands and torso from each other.

Once the microphone array was validated and its capability was visually confirmed, a set of gestures was presented and recorded. Chapter 7 shows how feature extraction allowed 3D traces of captured gestures to be visualised. This process was first tested using a corner reflector, which was designed to be approximately hand-sized. The use of corner reflector provided enhanced SNR and therefore greatly simplified the feature extraction. This allowed classification to work on a consistent and clear dataset. Traces were compared using DTW algorithm and classified using a 1NN classifier. This combination of DTW and 1NN was known to be an effective approach for classifying time-series. The result showed excellent gesture recognition with 97.5% of the tested gestures being classified correctly.

Finally, the same gestures were presented without using the corner reflector. The gestures were classified using the same process followed for the corner reflector. The accuracy of classification was lower than when the corner reflector was used, but remained above 80% for worst case gestures with overall average of 88.13%. This demonstrates the ability of the developed 2D microphone array to capture hand gestures in 3D with sufficient accuracy for automatic classification.

8.1 Recommendations for future work

This research was able to achieve high accuracy gesture recognition using a system developed from end to end. Other research reported in the literature has focused only on a part of the gesture recognition system or was limited to the development to a proof-of-concept that cannot be used for gesture recognition in a 3D space. The presented research provides a development of a fully functional 3D gesture recognition system and discusses all necessary considerations. 3D gesture recognition was successfully accomplished and demonstrates that ultrasound imaging shows potential for gesture recognition.

However, incorporating such broad development in a single project was achieved by making use of well established methods at each stage of the processes. While this allowed the system to be developed successfully, the efficiency and performance of the system may be improved by optimising various aspects of the complete system. One area that needs further consideration is the feature extraction, which was simplified by limiting the input to single hand gestures and improving the SNR using a corner reflector. By considering features that allow greater uncertainty and the presence of more than one moving target, robustness in practical application scenarios can be improved. One way in which this might be achieved is by avoiding the identification of peaks, but including a probabilistic indication of the possible location of targets in the feature vector. Also, the angular resolution of the beamforming array is less than ideal, as it cannot capture a detailed image of a hand. MVDR beamforming was attempted without success, but other beamforming methods to improve the angular resolution could still be considered.

Appendix A

Simulation of a Compensated CIC Filter

The following code plots the frequency response of a CIC filter as calculated in the Equation 4.15 along with the responses of compensation filters and compensated CIC filters for a comparison. The resulting graph is shown in the Figure 4.15.

```
import numpy as np
from numpy import sin
from numpy import abs
from numpy import pi
from numpy import log10

import pylab
from pylab import plot
from pylab import grid

D = 32
R = 32
M = 4

# Sinc and CIC filter response
SINC = lambda x : abs(sin(pi*x*D)/(pi*x))**M/(D**(M)) if (
    sin(pi*x)!=0) else 1
CIC = lambda x : abs(sin(pi*x*D)/sin(pi*x))**M/(D**(M)) if (
    sin(pi*x)!=0) else 1

# Constants used for CIC Compensation filter design
A = (1 - (D**-2)) / (1 - (2**-2))
B = (1 - (2*D)**-2) / (1 - (2**-4))

# 2nd order maximally flat cic compensation filter
a0 = M/-32 *A
```

```
a1 = 1-2*a0

COMPENSATOR_2nd = lambda x : abs(a0 + a1*np.exp(-2j*pi*x*R)
    + a0*np.exp(-4j*pi*x*R))

# 4th order maximally flat cic compensation filter
b0 = (2**-8) * M*A*(1/(2**3)*M*A + 1 - ((2**-2)*B))
b1 = -2**2 * b0 - 2**-5 *M*A
b2 = 1 - 2*b0 - 2*b1

COMPENSATOR_4th = lambda x : abs(b0 + b1*np.exp(-2j*pi*x*R)
    + b2*np.exp(-4j*pi*x*R) + b1*np.exp(-6j*pi*x*R) + b0*np.
    exp(-8j*pi*x*R))

# Compute the filter responses
w = np.arange(0,5000) * 0.51/5000

uncompensated_cic = np.array(map(CIC, w))
comp_2nd = np.array(map(COMPENSATOR_2nd,w))
comp_4th = np.array(map(COMPENSATOR_4th,w))

# Plot the responses
w = w*2*2400000
plot(w, 20*log10(uncompensated_cic))
plot(w, 20*log10(comp_2nd * uncompensated_cic))
plot(w, 20*log10(comp_4th * uncompensated_cic))
plot(w, 20*log10(comp_2nd), '--')
plot(w, 20*log10(comp_4th), '--')
grid('on')
pylab.xlim(0,80000)
pylab.ylim(-30,20)
pylab.show()
```

Appendix B

PDM Modulation

The following Python code modulates 40 kHz sine wave into PDM signal and store the output in the file named “output.bin”. The code stores 64 repeated bits and assumes 4.8 MHz sampling rate so the output is compatible with data from the microphone array.

```
import math
f = open("output.bin", 'wb')

integral = 0
output = 0
period = 4800000.0/40000

for i in range(480000):
        signal_in = math.sin(2*math.pi * i/period)

        # Feed back / Integrator
        integral += (signal_in - output)

        # Comparator
        if (integral >= 0):
                output = 1
                f.write(b"\xFF\xFF\xFF\xFF")
                f.write(b"\xFF\xFF\xFF\xFF")
        else:
                output = -1
                f.write(b"\x00\x00\x00\x00")
                f.write(b"\x00\x00\x00\x00")

f.close()
```

Appendix C

PDM Demodulation

The code reads bit array from a file and demodulates them using the compensated CIC filter designed in Section 4.4.

```
import numpy as np
import pylab as pl

def read_bitstream(file, length):
        """
        Read [length] samples from [file]
        File should be a binary with 64 bit streams of data
           interlaced and bit packed
        The function returns 64x[length] array. I.E. 64 bit
           stream of given length
        """

        bytes_read = np.fromfile(file, dtype=np.uint8, count
           =8*length)
        if(len(bytes_read) < length*8):
                print("The file prematurely ended")

        # Unpack the data into a bit stream, and
           deinterleave to 64 separate arrays
        bit_stream = np.unpackbits(bytes_read)
        bit_stream = np.reshape(bit_stream, (len(bit_stream)
           /64,64)).transpose()

        # Change all 0 to -1
        bit_stream = bit_stream.astype(np.int8)
        bit_stream[bit_stream==0] = -1
        return bit_stream

def CIC_filter(signal, order, delay, ratio):
```

```
        """Adapted from: http://stackoverflow.com/questions
           /14313510/how-to-calculate-moving-average-using-
           numpy"""

        # Apply cumulative sum
        for i in range(order):
                signal = np.cumsum(signal)
                                    # Keep it integer
                   arithematic

        # Decimate
        signal = signal[::ratio]

        dec_delay = delay/ratio
                            # Decimated delay
        for i in range(order):
                signal[dec_delay:] = signal[dec_delay:] -
                   signal[:-dec_delay]

        signal = 1.0*signal/delay**order
                   # Normalised

        return signal

def FIR_filter(signal):
        """Convolve the input signal with predefined fir
           compensation filter"""

        # 4th order fir filter D32 R32 M4
        fir_filter = np.array([0.0291259884834,
           -0.283007860184, 1.5077637434, -0.283007860184,
           0.0291259884834])
        return np.convolve(fir_filter, signal)[:len(signal)]

def main():
        # Read file
        f = open("output.bin", 'rb')
        bit_stream = read_bitstream(f, 1024*256)
        f.close()

        # Choose the microphones to read by index
        mic_index = np.array([0])

        # Apply filters
        sig_demod = np.apply_along_axis(CIC_filter, 1,
           bit_stream[mic_index,:], 4, 32, 32)
```

```
        sig_demod = np.apply_along_axis(FIR_filter, 1,
            sig_demod)

        # Plot sampled siusoid as a reference
        pl.rc('axes', labelsize=25)
        pl.subplot(2,1,1)
        pl.xlabel("40 kHz sampled sinusoid")
        pl.plot(np.sin(2.0*np.pi/3.75*np.array(range(64))))

        # Plot time domain graph
        pl.subplot(2,1,2)
        pl.xlabel("Demodulated 40 kHz sinusoid")
        for i, mic_ind in enumerate(mic_index):
                pl.plot(sig_demod[i][:64])
        pl.show()

        # Plot frequency domain graph
        for i, mic_ind in enumerate(mic_index):
                sig_short = sig_demod[i][:4096]
                pl.plot(np.fft.fftfreq(len(sig_short),
                    32.0/4800000), np.abs(np.fft.fft(
                    sig_short)))
        pl.xlabel("Frequency (Hz)")
        pl.show()

if __name__ == '__main__':
        main()
```

Appendix D

The gain of phase shift beamformer

The following code simulates the gain of a planar phase shift beamformer and plots its 3D gain pattern.

```
import numpy as np
from numpy import exp
from numpy import pi
from numpy.linalg import norm

import pylab as plt
from pylab import plot
from pylab import grid

from mpl_toolkits.mplot3d import Axes3D
from matplotlib import cm
from matplotlib.ticker import LinearLocator,
    FormatStrFormatter

f = 40000                   # Nominal frequency
v = 340                     # Speed of sound in air
d = 0.00425                 # Distance between microphones

fs = 4800000.0              # Sampling rate

m = 8                       # Width of the mic array
n = 8                       # Height of the mic array

def gain(phi, delay):
        """Calculate the gain of 1 dimensional mic array."""
        gain = 0
        for i in range(0,m):
                pm = np.array([(i-(m-1.0)/2)*d, 0])
```

```
                        #position of mic
                ps = np.array([np.cos(phi), np.sin(phi)])
                        #position of source
                gain += 1.0/m * norm(ps)/norm(pm-ps) *exp(-1
                    j*2*pi*f * ((norm(ps)-norm(pm-ps))/v +
                    delay*i) ) #near-field omni-directional
        return abs(gain)

def gain2D(phi, theta, xdelay, ydelay):
        """Gain of 2 dimensional mic array"""
        gain = 0
        for i in range(0,n):
                for j in range(0,m):
                        pm = np.array([(j-(m-1.0)/2)*d, (i-(
                            n-1.0)/2)*d, 0]) #position of the
                             mics
                        ps = 0.3*np.array([np.cos(phi)*np.
                            cos(theta), np.cos(phi)*np.sin(
                            theta), np.sin(phi)]) #position
                            of the source
                        gain += 1.0/(m*n) * norm(ps)/norm(pm
                            -ps) *exp(-1j*2*pi*f * ((norm(ps)
                            -norm(pm-ps))/v +xdelay*j +ydelay
                            *i) ) #near-field omni-
                            directional
        return abs(gain)


def gain_farfield(phi, delay):
        gain = 0
        for i in range(0,m):
                diff = (i-(m-1)/2)*d*np.cos(phi)
                distance = 1
                gain += 1.0/m * 1/distance *exp(-1j*2*pi*f *
                    (diff/v+delay*i))

        return abs(gain)


farfield = []
gain_1d = []
delay = 0.0

points = 51

phi = np.array(range(0,points))
```

```
theta = np.array(range(0,points))

gains = np.zeros(shape=(points,points))
x_axis = np.zeros(shape=(points,points))
y_axis = np.zeros(shape=(points,points))
z_axis = np.zeros(shape=(points,points))

##### 3D plotting of directivity #####
print "finding points"
for i in phi:
        for j in theta:
                gains[i][j] = (40+10*np.log10(gain2D(i/(
                    points-1.0)*pi, j/(points-1.0)*pi
                    ,0.0/4800000,0.0/4800000)))
                x_axis[i][j] = gains[i][j] * np.cos(i/(
                    points-1.0)*pi) * np.cos(j/(points-1.0)*
                    pi)
                y_axis[i][j] = gains[i][j] * np.cos(i/(
                    points-1.0)*pi) * np.sin(j/(points-1.0)*
                    pi)
                z_axis[i][j] = gains[i][j] * np.sin(i/(
                    points-1.0)*pi)
        print 100*i/(points-1.0),"%"
print(np.amin(gains))

fig = plt.figure()
ax = fig.gca(projection='3d')
surf = ax.plot_surface(x_axis, y_axis, z_axis, rstride=1,
    cstride=1, facecolors=cm.Spectral_r(gains/np.amax(gains))
    , linewidth=0, antialiased=True)

ax.set_xlabel('x')
ax.set_ylabel('y')
ax.set_zlabel('z')

ax.zaxis.set_major_locator(LinearLocator(10))
ax.zaxis.set_major_formatter(FormatStrFormatter('%.02f'))

mapped = cm.ScalarMappable(cmap=cm.jet)
mapped.set_array(gains)
fig.colorbar(mapped, shrink=0.5, aspect=5)

grid('on')
plt.show()
```

Appendix E

Beamformer implementation

The following code applies phase shift beamforming to a planar array and stores each 3D images' maxima to a text file. The points stored in the text file can be read and plotted to produce a trace of the gesture. The code also includes an implementation of MVDR beamformer that was tested, but not used for the final gesture recognition.

```
import numpy as np
import pylab as pl
from mpl_toolkits.mplot3d import Axes3D

import DAS_beamforming as das

def get2DSteeringVector(theta, phi, m, n):
        """Compute steering vector for a 2D array"""

        d = 0.00425                                    # distance
           between the elements
        wavelen = 340.0/40000    # $\lambda$

        sound_dir = (np.sin(theta), np.cos(theta))
        sound_dir = sound_dir/np.linalg.norm(sound_dir) #
           Unit length vector in theta direction

        steering_vec = np.zeros((m,n), dtype=np.complex_)
        for i in range(m):
                for j in range(n):
                        mic_vec = (i -(m-1)/2.0, j -(n-1)
                           /2.0)
                        steering_vec[i][j] = np.exp(-2j*np.
                           pi*d*np.dot(mic_vec, sound_dir)/
                           wavelen *np.sin(phi)) # steering
                           direction as in equ 3.8
```

```
        return steering_vec

def PhaseShift3D(signal, phi, theta):
        """
        Apply phase shift beamforming on a 2D array.
        """

        # First, figure out the weight vector for DAS which
            is equal to steering vector
        w = get2DSteeringVector(theta, phi, 8, 8)
        w = w.reshape(64)

        signal = signal.astype(complex)
        for ind in range(len(signal)):
                signal[ind] = w[ind] * signal[ind]
        summed = np.sum(signal,0)/len(signal)
        return summed

def MVDR3D(signal, phi, theta):
        """
        Apply MVDR beamforming on a 2D array.
        It was not used for the final processing.
        """

        # Compute auto covariance matrix Rxx and its inverse
            - only need it once per data set
        Rxx = np.corrcoef(signal)
        Rxx_inv = np.linalg.inv(Rxx)

        # Compute steering vector
        steering_vec = np.atleast_2d(get2DSteeringVector(
            theta, phi, 8, 8).reshape(64)).T

        # Compute MVDR weighting
        try:
                filter_w = np.dot(Rxx_inv, steering_vec)
                filter_w = filter_w.dot(np.linalg.inv(
                    steering_vec.T.conj().dot(filter_w)))
        except:
                print("Non invertible matrix")
                filter_w = steering_vec

        signal = signal.astype(complex)
        for ind in range(len(signal)):
                signal[ind] =  filter_w[ind] * signal[ind]
        summed = np.sum(signal,0)
```

```
        return summed

def create_image_3D(delay_func, sig_demod):
        # Get reflected signal, going around the scene in
            circle

        incident_angle = np.arange(-80, 81, 6)/180.0 * np.pi
        azim_angle = np.arange(0, 181, 12)/180.0 * np.pi
        complex_sinusoid = np.exp(2j*np.pi*40000.0/150000*np
            .array(range(10)))


        signal3d = np.empty((len(incident_angle), len(
            azim_angle), 1024))

        for t, theta in enumerate(azim_angle):
                for p, phi in enumerate(incident_angle):
                        beamformed = delay_func(sig_demod,
                            phi, theta)
                        # Filtering out the 40kHz signal
                            only
                        signal3d[p][t] = abs(np.correlate(
                            beamformed, complex_sinusoid,
                            mode='same'))
        max_polar = list(np.unravel_index(signal3d.argmax(),
             dims=(signal3d.shape)))

        max_polar[2] = (max_polar[2])/2/150000.0*340 #
            Normalise distance
        max_cart = max_polar[2]*np.sin(incident_angle[
            max_polar[0]])*np.cos(azim_angle[max_polar[1]]),
            max_polar[2]*np.sin(incident_angle[max_polar[0]])
            *np.sin(azim_angle[max_polar[1]]), max_polar[2]*
            np.cos(incident_angle[max_polar[0]])
        print(max_cart)

        # Store the points in a txt file
        outfile = open("traces.txt", 'a')
        np.savetxt(outfile, np.array(max_cart)[None, :],
            delimiter=' ')
        outfile.close()

def main():
        f = open("di_read.dat", 'rb')

        f.seek(60) # Skip the metadata
```

```
        # Reference the microphones in correct order
        mic_ref = np.array([0, 32, 1, 33, 17, 49, 16, 48])
        mic_ref = np.array([mic_ref+6-2*x for x in range(4)
            ]+ [mic_ref+14-2*x for x in range(4)])
        mic_ref = mic_ref.reshape(64)

        sig_length = 200
        i = 0
        while 1:
                bit_stream = das.read_bitstream(f, 1024*32)
                    # Note: 1024*32 = 1 frame
                if bit_stream is None:
                                            # End if EOF is
                    reached
                        break
                sig_demod = np.apply_along_axis(das.
                    CIC_filter, 1, bit_stream[mic_ref,:], 4,
                    32, 32)
                sig_demod = np.apply_along_axis(das.
                    FIR_filter, 1, sig_demod)
                sig_demod[:, :sig_length] = 0
                create_image_3D(PhaseShift3D, sig_demod)
                f.seek(8*32*1024*4,1)    # Skip 4 frames
        f.close()


if __name__ == '__main__':
        main()
```

Appendix F

Classification

This code performs classification based on DTW and kNN on the traces produced by Appendix E. The code reads text files that stores the gestures, organised by folders according to the kind of gesture it represents. A part of the data that was chosen as the test cases are taken out and get classified to a gesture using the rest of the data as the training set. The based on the result from the test cases, the probability of correctly classifying a gesture is printed for each gesture.

```
# DTW + KNN (k=1)

import numpy as np
from numpy.linalg import norm
import matplotlib.pyplot as pl
from mpl_toolkits.mplot3d import Axes3D
from random import random

def DTW_distance(s1, s2):
        """
        Simple DTW distance calculator
        Exact Python implemnetation of the algorithm in
           Wikipedia
        """

        DTW_grid={}

        for i in range(len(s1)):
                DTW_grid[(i, -1)] = float('inf')
                for i in range(len(s2)):
                        DTW_grid[(-1, i)] = float('inf')
                        DTW_grid[(-1, -1)] = 0

        for i in range(len(s1)):
                for j in range(len(s2)):
```

```
                        dist= norm(s1[i]-s2[j])**2
                        DTW_grid[(i, j)] = dist + min(
                            DTW_grid[(i-1, j)],DTW_grid[(i, j
                            -1)], DTW_grid[(i-1, j-1)])

        return np.sqrt(DTW_grid[len(s1)-1, len(s2)-1])

def main():

        categories = ["Left", "Right", "Up", "Down", "Toward
            ", "Away", "Clock", "AntiClock"]
        testcase = range(1,6) # Testcases for k-fold
            validation

        # Read trace templates with respective labels
        templates = []
        labels = []
        for category in categories:
                filename_rel = category + r"/traces (" #
                    Read the training data stored in
                    respectively named folders
                for i in range(1,21):
                        if i in testcase:       # Don't
                            include in the template if it is
                            a test case
                                continue
                        filename = filename_rel+str(i)+")"
                        prev_file = open(filename+".txt", 'r
                            ')
                        data = np.genfromtxt(prev_file,
                            delimiter=' ')

                        #Filter outliers, happens when the
                            object is out of the scene
                        data = data[np.where(abs(data[:,0])
                            < 0.35)]
                        data = data[np.where(abs(data[:,1])
                            < 0.35)]
                        data = data[np.where(data[:,2] >
                            0.2)]

                        templates.append(data)
                        labels.append(category)


        for category in categories:
```

```
                score = 0
                for i in testcase:
                        # Read the test cases in each
                            category
                        test = np.genfromtxt("C:\Users\
                            Hankyu\Documents\Workspace\
                            Masters data\\"+ category +"\
                            di_read ("+str(i)+").txt",
                            delimiter=' ')
                        test = test[np.where(abs(test[:,0])
                            < 0.35)]
                        test = test[np.where(abs(test[:,1])
                            < 0.35)]
                        test = test[np.where(test[:,2] >
                            0.2)]

                        # Find DTW distance to every
                            template
                        scores = np.zeros(len(templates))
                        for i in range(len(templates)):
                                scores[i] = DTW_distance(
                                    test, templates[i])

                        # Check if the nearest template is
                            the right category
                        if labels[np.argmin(scores)] ==
                            category:
                                score+=1
                        else:
                                # Show which category it was
                                    assigned to, if it was
                                    matched wrong
                                print(labels[np.argmin(
                                    scores)])
                print(category, 1.0*score/len(testcase))
                        # Show the accuracy for each
                    category

if __name__ == '__main__':
        main()
```

List of References

Abdelnasser, H., Youssef, M. and Harras, K.A. (2015). Wigest: A ubiquitous wifi-based gesture recognition system. *CoRR*, vol. abs/1501.04301.
Available at: http://arxiv.org/abs/1501.04301

Adib, F. and Katabi, D. (2013 August). See through walls with wifi! *SIGCOMM Comput. Commun. Rev.*, vol. 43, no. 4, pp. 75–86. ISSN 0146-4833.
Available at: http://doi.acm.org/10.1145/2534169.2486039

Arbabian, A., Callender, S., Kang, S.-Y., Rangwala, M. and Niknejad, A.M. (2013). A 94 ghz mm-wave-to-baseband pulsed-radar transceiver with applications in imaging and gesture recognition. *Solid-State Circuits, IEEE Journal of*, vol. 48, no. 4, pp. 1055–1071.

Benesty, J., Chen, J. and Huang, Y. (2008). *Microphone array signal processing*, vol. 1. Springer Science & Business Media.

Berman, S. and Stern, H. (2012). Sensors for gesture recognition systems. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, vol. 42, no. 3, pp. 277–290.

Brown, A.D. (2012). *Electronically Scanned Arrays*. Synthesis Lectures on Antennas. Morgan & Claypool Publishers. ISBN 9781598291827.
Available at: https://books.google.co.za/books?id=xYBfAQAAQBAJ

Capon, J., Greenfield, R.J. and Kolker, R.J. (1967 Feb). Multidimensional maximum-likelihood processing of a large aperture seismic array. *Proceedings of the IEEE*, vol. 55, no. 2, pp. 192–211. ISSN 0018-9219.

Capps, C. (2001). Near field or far field? *Electrical Design News*, vol. 46, p. 18.

Cassinelli, Á., Perrin, S. and Ishikawa, M. (2005). Smart laser-scanner for 3d human-machine interface. In: *CHI '05 extended abstracts on Human factors in computing systems - CHI '05*, p. 1138. ACM Press, New York, New York, USA. ISBN 1595930027.
Available at: http://portal.acm.org/citation.cfm?doid=1056808.1056851

Chen, X. and Parks, T. (1987 Feb). Design of fir filters in the complex domain. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 35, no. 2, pp. 144–153. ISSN 0096-3518.

Cheng, H., Dai, Z., Liu, Z. and Zhao, Y. (2016*a*). An image-to-class dynamic time warping approach for both 3d static and trajectory hand gesture recognition. *Pattern Recognition*, vol. 55, no. Supplement C, pp. 137 – 147. ISSN 0031-3203. Available at: `http://www.sciencedirect.com/science/article/pii/S0031320316000157`

Cheng, H., Yang, L. and Liu, Z. (2016 Sept*b*). Survey on 3d hand gesture recognition. *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 26, no. 9, pp. 1659–1673. ISSN 1051-8215.

Cho, K. and Ahmed, N. (1987 Sept). On a constrained lms dither algorithm. *Proceedings of the IEEE*, vol. 75, no. 9, pp. 1338–1340. ISSN 0018-9219.

Chryssomallis, M., Christodoulou, C. and Webster, J.G. (1999). *Antenna Radiation Patterns*. John Wiley & Sons, Inc. ISBN 9780471346081.
Available at: `http://dx.doi.org/10.1002/047134608X.W1204.pub2`

Condon, J.J. and Ransom, S.M. (2016). *Essential Radio Astronomy*. Princeton Series in Modern Observational Astronomy. Princeton University Press. ISBN 1400881161.
Available at: `http://press.princeton.edu/titles/10771.html`

Cooper, K., Dengler, R., Chattopadhyay, G., Schlecht, E., Gill, J., Skalare, A., Mehdi, I. and Siegel, P. (2008). A high-resolution imaging radar at 580 ghz. *Microwave and Wireless Components Letters, IEEE*, vol. 18, no. 1, pp. 64–66.

Dokmanic, I. and Tashev, I. (2014 May). Hardware and algorithms for ultrasonic depth imaging. In: *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*, pp. 6702–6706.

Dolecek, G.J. and Diaz-Carmona, J. (2011). *Applications of MATLAB in Science and Engineering*, chap. On Design of CIC Decimators. InTech. ISBN 978-953-307-708-6.
Available at: `http://www.intechopen.com/books/applications-of-matlab-in-science-and-engineering/on-design-of-cic-decimators`

Du, W. and Kirlin, R.L. (1991 May). Improved spatial smoothing techniques for doa estimation of coherent signals. *IEEE Transactions on Signal Processing*, vol. 39, no. 5, pp. 1208–1210. ISSN 1053-587X.

Fernandez-Vazquez, A. and Dolecek, G. (2012 Feb). Maximally flat cic compensation filter: Design and multiplierless implementation. *Circuits and Systems II: Express Briefs, IEEE Transactions on*, vol. 59, no. 2, pp. 113–117. ISSN 1549-7747.

Franco, E.E., Andrade, M.a.B., Adamowski, J.C. and Buiochi, F. (2011). Acoustic beam modeling of ultrasonic transducers and arrays using the impulse response and the discrete representation methods. *Journal of the Brazilian Society of Mechanical Sciences and Engineering*, vol. 33, no. 4, pp. 408–416. ISSN 1678-5878.

Gupta, S., Morris, D., Patel, S. and Tan, D. (2012). Soundwave: Using the doppler effect to sense gestures. In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '12, pp. 1911–1914. ACM, New York, NY, USA. ISBN 978-1-4503-1015-4.
Available at: `http://doi.acm.org/10.1145/2207676.2208331`

Harput, S. and Bozkurt, A. (2008). Ultrasonic phased array device for acoustic imaging in air. *IEEE sensors journal*, vol. 8, no. 11, pp. 1755–1762.

Haykin, S. (1991). *Adaptive filter theory*. Prentice-Hall information and system sciences series. Prentice Hall. ISBN 9780130132369.
Available at: `https://books.google.co.za/books?id=E5hTAAAAMAAJ`

Huang, D., Nandakumar, R. and Gollakota, S. (2014). Feasibility and limits of wi-fi imaging. In: *Proceedings of the 12th ACM Conference on Embedded Network Sensor Systems*, SenSys '14, pp. 266–279. ACM, New York, NY, USA. ISBN 978-1-4503-3143-2.
Available at: `http://doi.acm.org/10.1145/2668332.2668344`

Hunt, A., Tillery, C. and Wild, N. (2001). Through-the-wall surveillance technologies. *Corrections Today*, vol. 63, no. 4, p. 132.

Johnson, R. and Jasik, H. (1993). *Antenna Engineering Handbook*. Electronics Electrical Engineering. McGraw-Hill. ISBN 9780070323810.

Kellogg, B., Talla, V. and Gollakota, S. (2014 April). Bringing gesture recognition to all devices. In: *11th USENIX Symposium on Networked Systems Design and Implementation (NSDI 14)*, pp. 303–316. USENIX Association, Seattle, WA. ISBN 978-1-931971-09-6.
Available at: `https://www.usenix.org/conference/nsdi14/technical-sessions/presentation/kellogg`

Kino, G. (1987). *Acoustic waves: devices, imaging, and analog signal processing*. Prentice-Hall Signal Processing Series. Prentice Hall PTR. ISBN 9780130030474.
Available at: `https://books.google.co.za/books?id=hcsYAQAAIAAJ`

Knowles Electronics (2014). Digital zero-height sisonic™ microphone with multi-mode and ultrasonic support. Rev. A.

LearningaboutElectronics (2016). How to build a simple function generator with an lm324 op amp chip.
Available at: `http://www.learningaboutelectronics.com/Articles/Function-generator-circuit.php`

Lyon, R.F. (2007 January). Wave diffraction 4lambda slit.
Available at: `https://commons.wikimedia.org/wiki/File:Wave_Diffraction_4Lambda_Slit.png`

Lyons, R.G. (2005). Understanding cascaded interator-comb filters.
Available at: `http://www.embedded.com/design/configurable-systems/4006446/Understanding-cascaded-integrator-comb-filters`

McCowan, I. (2001). Microphone arrays: A tutorial. *Queensland University, Australia*, pp. 1–38.

Mitsa, T. (2010). *Temporal Data Mining*. 1st edn. Chapman & Hall/CRC. ISBN 1420089765, 9781420089769.

Mladenov, V., Karampelas, P., Tsenov, G. and Vita, V. (2011). Approximation formula for easy calculation of signal-to-noise ratio of sigma-delta modulators. *ISRN Signal Processing*, vol. 2011.

Moebus, M. and Zoubir, A.M. (2007 April). Three-dimensional ultrasound imaging in air using a 2d array on a fixed platform. In: *Acoustics, Speech and Signal Processing, 2007. ICASSP 2007. IEEE International Conference on*, vol. 2, pp. II–961–II–964. ISSN 1520-6149.

Molchanov, P., Gupta, S., Kim, K. and Pulli, K. (2015 May). Multi-sensor system for driver's hand-gesture recognition. In: *Automatic Face and Gesture Recognition (FG), 2015 11th IEEE International Conference and Workshops on*, vol. 1, pp. 1–8.

Monzingo, R.A., Haupt, R.L. and Miller, T.W. (2011 jan). *Introduction to Adaptive Arrays*. Institution of Engineering and Technology. ISBN 9781891121579.
Available at: `http://digital-library.theiet.org/content/books/ew/sbew046e`

Mucci, R.A. (1984). A comparison of efficient beamforming algorithms. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 32, no. 3, pp. 548–558. ISSN 00963518.

Müller, M. (2007). *Information retrieval for music and motion*, vol. 2. Springer.

Nandakumar, R., Kellogg, B. and Gollakota, S. (2014). Wi-fi gesture recognition on existing devices. *CoRR*, vol. abs/1411.5394.
Available at: `http://arxiv.org/abs/1411.5394`

Ng, A. and Swanevelder, J. (2011). Resolution in ultrasound imaging. *Continuing Education in Anaesthesia, Critical Care & Pain*, vol. 11, no. 5, pp. 186–192.

Park, S. and Motorola, I. (1990). *Principles of sigma-delta modulation for analog-to-digital converters*. [Phoenix, Ariz.] : Motorola. APR8/D.

Perrin, S., Cassinelli, A. and Ishikawa, M. (2004). Gesture recognition using laser-based tracking system. In: *Proceedings - Sixth IEEE International Conference on Automatic Face and Gesture Recognition*, pp. 541–546. ISBN 0769521223.

Plouffe, G. and Cretu, A.M. (2016 Feb). Static and dynamic hand gesture recognition in depth data using dynamic time warping. *IEEE Transactions on Instrumentation and Measurement*, vol. 65, no. 2, pp. 305–316. ISSN 0018-9456.

Przybyla, R., Tang, H.-Y., Shelton, S., Horsley, D. and Boser, B. (2014 Feb). 12.1 3d ultrasonic gesture recognition. In: *Solid-State Circuits Conference Digest of Technical Papers (ISSCC), 2014 IEEE International*, pp. 210–211. ISSN 0193-6530.

Pu, Q., Gupta, S., Gollakota, S. and Patel, S. (2013). Whole-home gesture recognition using wireless signals. In: *Proceedings of the 19th annual international conference on Mobile computing & networking*, pp. 27–38. ACM.

Reddy, V., Paulraj, A. and Kailath, T. (1987 Jul). Performance analysis of the optimum beamformer in the presence of correlated sources and its behavior under spatial smoothing. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 35, no. 7, pp. 927–936. ISSN 0096-3518.

Riu, J. and Royo, S. (2013). Lidar imaging with on-the-fly adaptable spatial resolution. In: *Proceedings of SPIE - The International Society for Optical Engineering*, vol. 8897, p. 88970N. ISBN 9780819497666. ISSN 0277786X.
Available at: `http://www.scopus.com/inward/record.url?eid=2-s2.0-84890036188{&}partnerID=tZOtx3y1`

Shung, K.K. and Zippuro, M. (1996). Ultrasonic transducers and arrays. *IEEE Engineering in Medicine and Biology Magazine*, vol. 15, no. 6, pp. 20–30.

Stutzman, W.L. (1998). Estimating directivity and gain of antennas. *IEEE Antennas and Propagation Magazine*, vol. 40, no. 4, pp. 7–11. ISSN 10459243.
Available at: `http://ieeexplore.ieee.org/xpls/abs{%}7B{_}{%}7Dall.jsp?arnumber=730532`

Tang, M.-C., Wang, F.-K. and Horng, T.-S. (2015). Human gesture sensor using ambient wireless signals based on passive radar technology. *ILO*, vol. 90, p. 0.

Tashev, I. (2009). *Sound Capture and Processing: Practical Approaches*. Wiley. ISBN 9780470319833.
Available at: `https://books.google.co.za/books?id=o8-4DAEACAAJ`

TheAverageBody.com (2014). Average hand size.
Available at: `http://www.theaveragebody.com/average_hand_size.php`

Tyson, D. (2015). Google introduces project: Soli - radar based touchless input control for wearables - ausdroid.
Available at: `http://ausdroid.net/2015/05/30/google-introduces-project-soli-radar-based-touchless-input-control-for-wearables/`

Van Veen, B. and Buckley, K.M. (1997). Beamforming techniques for spatial filtering. *Digital Signal Processing Handbook*, pp. 61–1.

van Willigen, D.M., Mostert, E., Pertijs, M. *et al.* (2014). In-air ultrasonic gesture sensing with mems microphones. In: *SENSORS, 2014 IEEE*, pp. 90–93. IEEE.

Wang, Y. and Reiss, J.D. (2012 Apr). Time domain performance of decimation filter architectures for high resolution sigma delta analogue to digital conversion. In: *Audio Engineering Society Convention 132*.
Available at: `http://www.aes.org/e-lib/browse.cfm?elib=16286`

Wojtczuk, P., Armitage, A., Binnie, T.D. and Chamberlain, T. (2012). Recognition of simple gestures using a pir sensor array. *Sensors & Transducers*, vol. 14, no. 1, pp. 83–94.

Xi, X., Keogh, E., Shelton, C., Wei, L. and Ratanamahatana, C.A. (2006). Fast time series classification using numerosity reduction. In: *In ICML'06*, pp. 1033–1040.

Zhao, C., Chen, K.-Y., Aumi, M.T.I., Patel, S. and Reynolds, M.S. (2014). Sideswipe: detecting in-air gestures around mobile devices using actual gsm signal. In: *Proceedings of the 27th annual ACM symposium on User interface software and technology*, pp. 527–534. ACM.

Zoltowski, M.D. (1988 Jun). On the performance analysis of the mvdr beamformer in the presence of correlated interference. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 36, no. 6, pp. 945–947. ISSN 0096-3518.